WIN A LYNX COMPUTER

...LING COMPUTER PROJECTS MAGAZINE DECEMBER 1983

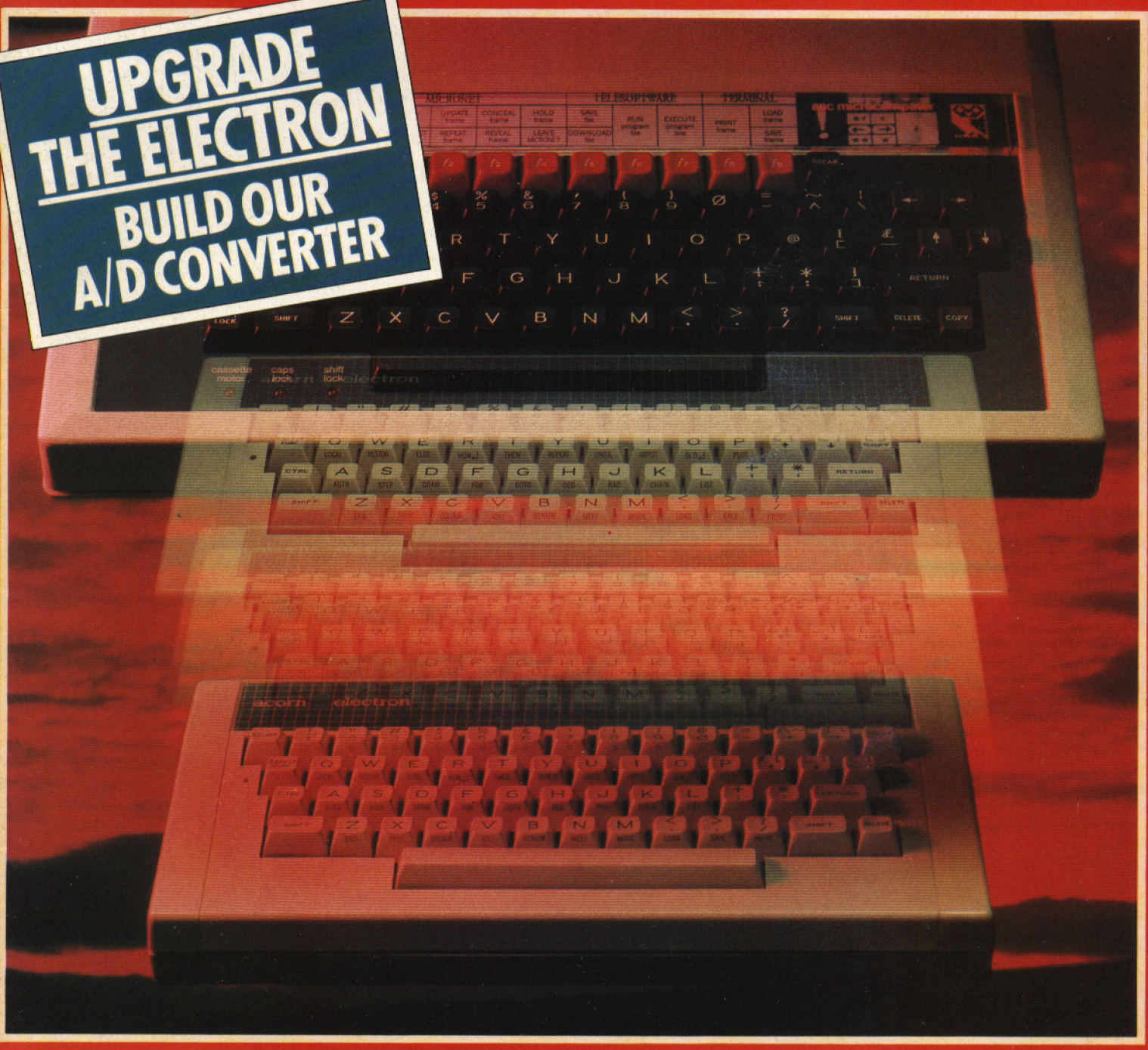
ELECTRONICS & COMPUTING

MONTHLY AN EMAP PUBLICATION

Germany D5.80
U.S.A. $2.95
85p

UPGRADE THE ELECTRON

BUILD OUR A/D CONVERTER



BUILD YOUR OWN 16 BIT COMPUTER

TOP MICRO'S SOFTWARE ON TRIAL • Z80 AND 6502 VECTORS

SPECTRUM TAPE BACKUP • BBC SIDEWAYS RAM PROJECT

Electronics & Computing Monthly
Scriptor Court,
155 Farringdon Road,
London,
EC1R 3AD

EDITORIAL
01-833-0846
Editor
Gary Evans

Assistant Editor
William Owen

Administration
Liz Gregory

ADVERTISING
01-833-0531

Manager
Claire Fullerton

Executive
Richard Jansz

Classified
Andrew King

PRODUCTION
01-833-0531

Design
Pat Haylock

Make-up
Time Graphics

Publisher
Alfred Rolington

Distribution
EMAP
National Publications

Published By
EMAP Business And
Computer Publications

Printed by
Riverside Press,
England

Subscriptions
Electronics & Computing Monthly,
(Subscription Department),
Competition House,
Farndon Road,
Market Harborough,
Leicestershire.

ABC
MEMBER OF THE AUDIT BUREAU OF CIRCULATION

BBC Sideways RAM 18
A natural progression from our BBC EPROM Programmer. This project does away with the problems associated with debugging programs written straight to EPROM.

Spectrum Tape Back-up 24
Following the information presented in this feature, you'll be able to make back-up copies of any valuable Spectrum software.

SAT 16 30
The start of a major new *E&CM* series that describes a dual processor 16 bit system based around the 68000 and 68701 devices.

Micro Graphic Techniques 35
Mike James continues his series on computer graphics with a look at the important area of controlling sprites.

Interface 2 38
A closer look at Sinclair's latest Spectrum add-on previewed in the last issue.

Electron A/D Board 40
The Electron has been widely described as a stripped down BBC micro. This month, in the first of a series of projects, we present a design for a 4 channel A/D converter that makes up for one of the computer's shortcomings.

The BBC MOS 50
Of the 32K of firmware on the BBC micro, half is devoted to the Machine Operating System. We take a look at the MOS and explain how it can be used both by BASIC and Machine Code programmers.

Currah's μSpeech reviewed 56
By making use of a ULA, Currah have managed to produce a sub £30 speech synthesiser featuring a text-to-speech interpreter. Gary Herman evaluates the device.

ZX Disco Light Controller 59
This month we describe some software for the controller from elementary on/off routines to a sophisticated machine code dimming routine.

Software Selection 64
News of some of the software available for the Spectrum, Dragon, Oric and BBC/Electron. With the festive season approaching the accent is on games.

Circuit Tips 67
A new occasional feature. This month a novec Pulse Recovery circuit.

Using Z80 and 6502 Vectors 71
In the first of a two part feature, Adam Denning introduces Vectors and begins to explain their use.

Hi-res computer's A/D Board 75
Construction and calibration information for this the latest card for our popular 6809 system.

Speech Synthesis Systems 83
Ian Campbell rounds off his look at speech synthesis systems with a look at National's Digitalker and at the techniques of LPC as used by the likes of Texas and National.

Single Chip Microcontroller 89
This month's instalment concludes the description of the features provided by the 68705.

6502 Processor Card 94
A low cost and easy to use 6502 processor card that can be expanded to form a sophisticated disk based system.

Book and Software Reviews 102
Harry Fairhead takes his regular look at recently published books and software.

Comment 9
Subscriptions 9
News 13
New Products 14
Letters 80
Next Month 96

Reader Services
PCB Service 57
Book Service 101

See Page 70 for details of this month's free to enter competition with a Lynx 48K computer as the prize.

Electronics & Computing Monthly is normally published on the 13th day of each month

 Subscription rates: UK £10.70 incl. post. For overseas rates apply to Subscription Dept., Competition House, Farndon Road, Market Harborough, Leics. Back issues available from: EMAP National Publications (*E&CM* Back Numbers), Bretton Court, Peterborough, PE3 8DZ. Phone 0733 264666.

EDITORIAL

At the moment, any conversation with those involved in the supply of components for the amateur electronic market tends to veer towards gloom and doom. The same can be said of discussions with people working for the popular electronic press.

The conventionally held view in both these camps is that people who a few years ago would adopt Electronics as their hobby interest are today attracted towards the glamour of home computers as their leisure interests. With this negative view from the trade there is a danger of a 'chicken and egg' situation in which both the retail and publishing arms of the Electronic industry gradually remove themselves from the amateur market and thus contribute to the decline in this area.

Personally, as one who has been involved in various aspects of the amateur electronics business for a number of years, I would be extremely sad to see the decline of the hobby electronics market. For this reason I was particularly pleased to see that BBC School Radio has produced two series of programmes specifically aimed at introducing electronics to primary and secondary school pupils.

The first of these programmes is 'Junior Electronics' and is aimed at children in the 9 to 12 year old age bracket. The BBC has collaborated with the Microelectronics Education Programme and the Department of Industry during the production of the series. The result is that a kit of electronic bits and pieces has been assembled to accompany the series. This means that pupils are offered the chance of tackling practical projects.

The second series is aimed at secondary school children taking 'O' level or 'O' grade CSE examinations. The series forms an introduction to basic microelectronics and leads to an introduction of the ideas of control technology. Again the series is accompanied by a kit of parts that allows the principles developed during the broadcasts to be further reinforced through practical work.

This series is called 'Microtechnology' and the ten programmes begin transmission on Friday January 20 at 10.45 a.m. ('Junior Electronics' starts on Tuesday, February 28 at 2.20 p.m. – both programmes go out on Radio 4 VHF).

Another new series from BBC Schools is 'Using Your Computer' which began transmission on November 1. This is aimed at primary and middle school children in the 9-12 year-old bracket. The aim of this production is to introduce children *and* teachers in the schools to the computers supplied under the DoI 'Computers in Schools' scheme – the BBC Micro, Spectrum and the Link 480Z.

The series is designed for recording and, used in this fashion, allows the radio broadcasts to be synchronised with the computer software that accompanies the broadcasts.

This first series introduces pupils to the computers themselves but plans are well advanced for using a similar format to teach subjects ranging from languages to astronomy.

The BBC may be accused of being slightly slow off the mark with this sort of material but these programmes, and others close to completion, mean that the BBC now have a firm commitment to the teaching of new technology.

It's also refreshing to see that the Corporation are not neglecting to introduce the fundamentals of electronics to school children and to emphasise that an understanding of electronics is essential if computer and control systems are to be put to the best use – A philosophy that Electronics & Computing has been promoting since our first issue over two years ago.

Making An Exhibition

A few days after this issue is due to appear on the news stands the Breadboard '83 exhibition is due to start at London's Cunard International Hotel. The exact dates of the show are from Friday November 25th to Sunday the 27th. Opening hours are from 10 a.m. to 6 p.m. on the first two days and from 10 a.m. to 4 p.m. on the Sunday.

Electronics & Computing will be at the show and we hope that any of you who can get to Breadboard will make a point of visiting our stand.

Gary Evans.

Spot The Electron

Most readers will be aware that the Acorn Electron was launched in August of this year, but very few will have had the chance to get their hands on what was predicted to be 'the hit micro of 1983'. Indeed, if anyone has actually seen an Electron in the shops we would like to hear about it!

There are currently two schools of thought regarding the fate of the Electron. The first, the official view, is that the machine is disappearing from the shelves as fast as they are stocked. W. H. Smith, who have signed a special deal with Acorn giving almost exclusive marketing rights, say that they have run out of initial supplies after 'phenomenal' demand, and are now awaiting new stock. The high street stationers would not, however, give a date as to when new supplies would be available, and nor would they say how many machines have been sold to date.

The second, 'unofficial' view is based on the experience of some software suppliers who are unable to obtain sample machines from the manufacturer, and of well known High Street dealers who are being supplied with the Electron at the rate of only one unit per month, despite having been on a waiting list for one year.

This situation has fueled reports that the Electron has been withdrawn following the discovery of a design fault. *E&CMs* own investigations elicited a flat denial from Acorn's PR company, but Acorn's Marketing Manager did say that in fact 100 machines had been withdrawn from warehouse stock when a small electrical fault was discovered. Apparently the insulation between keyboard and PCB was inadequate, and there was the possibility that if the keys were pressed too firmly a short would occur. Acorn insist that only this small batch was affected, and that these were being repaired in house.

So is there still a chance that the Electron will become the hit micro of 1983, or will this excellent machine have to wait until 1984 to make it's mark? Estimates of Electron production differ widely between 140,000 and only 8000. If the latter figure is accurate (and the supply problems faced by dealers suggest that it may be) then there is no chance of Acorn meeting the Christmas demand and a January relaunch will be on the cards.

Goings On At Oric

A recent report in Marketing Week indicated that Oric were intending to phase out production of their 16K machine and concentrate fully on the 48K Oric 1, and, in order to rid itself of remaining stocks, was going to sell off the 16K at rock bottom prices over the Christmas period. The Company itself has a different story to tell. Oric deny that they have any intention of discontinuing the 16K machine while a demand for it persists. But Oric are reducing the price of the 16K from £100 to a mere £80 for those machines sold through Tansoft. This would appear to be an excellent offer and means that the Oric will undercut the Spectrum, the closest comparible machine, by a full £20.

Much pooh poohing of the British computer press' bad reception of the Oric accompanied the French award of 'Home Computer of the Year' to this very same machine. The Oric has been very successful in Europe, notably so in France (where it is one of the few machines capable of driving SECAM, the local TV standard).

A seal of approval from the French press, and a price cut into the bargain will do Oric no harm in the British market. The company's manoeuvring on the financial market has met with rather less success. In order to drum up capital Oric has attempted to go public (à la Acorn), by a devious route known as a reverse takeover. Unfortunately the Stock Exchange blocked Oric because the Company has only one year of trading behind it.

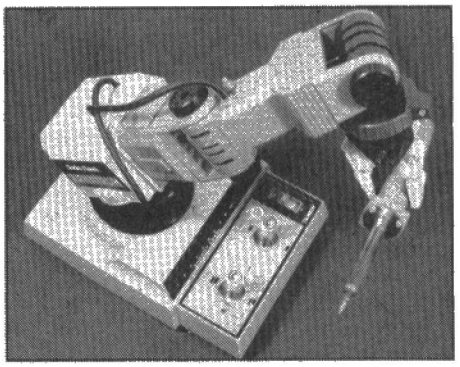
Aquarius Price Plummets

The computer price war is hotting up. Not only Oric, but also the Mattel Aquarius has been slashed in price (will Sinclair be forced to cut again?) It is now possible to buy a colour computer for a mere £50. The Aquarius has been spotted selling at this price in Argos Catalogue showrooms, and this represents a drop in price of 50% since the Aquarius first came on the market this summer. Surely something of a record?

This Is Customer Support

A constant complaint of home computer users is the total inadequacy of technical support for machines, and in particular peripherals. Certain manufacturers would be well advised to take a lesson from Hewlett Packard if they want to win a reputation for customer support and assistance. This US company recently opened a new centre entirely for this purpose at Winnersh, near Reading. The centre is responsible for maintaining, repairing and calibrating no less than 3/4 of HP's range of 7000 products, including personal and business computers.

Many instruments no longer in the catalogue are also maintained at the centre, because HP guarantees to service equipment for 5 years after it has been withdrawn. It is not uncommon for 30 year old wooden boxed instruments to come in from Harley Street for repair. The new centre is expected to handle £100 million of shipments in the first year, and is equipped to make 2200 instrument repairs a month and 1100 computer repairs.



Armatron

This monster holding a soldering iron between its teeth is the Tandy Armatron, a single motor driven robot arm controlled by two joysticks.

The arm is capable of rotary and lateral movement over 3 axes. It is probably the cheapest robot arm on the market, at only £25.00.

E&CM will be attempting to discover the possibilities of computer control of what is at the moment little more than a superior executive toy (the accuracy of movement leaves a lot to be desired), but more serious applications may be revealed.

80-Bus System News

Gemini Microcomputers have released their latest catalogue detailing the range of products which make up the Galaxy 80-bus microsystem. The philosophy behind this multiboard system is to allow an extremely flexible approach to system configuration, utilising the wide range of software available for the Z80 microcomputer. Users can either buy a complete system or build from a basic CPU board with a series of RAM, EPROM, disc controller, graphics and bus boards. Keyboards of various types, Winchester and floppy drives, networking systems, prototyping boards, power supplies etc. are also available

Henry's of Edgware Road, a major supplier of the Galaxy range, have at the same time announced their entry into the software market with the introduction of MDIS, an intelligent disassembler which can be used on all CP/M based machines. MDIS can be used to take apart CP/M machine code programs for examination or modification. Using the correct mnemonics in each case it will disassemble both Z80 and 8080 machine codes.

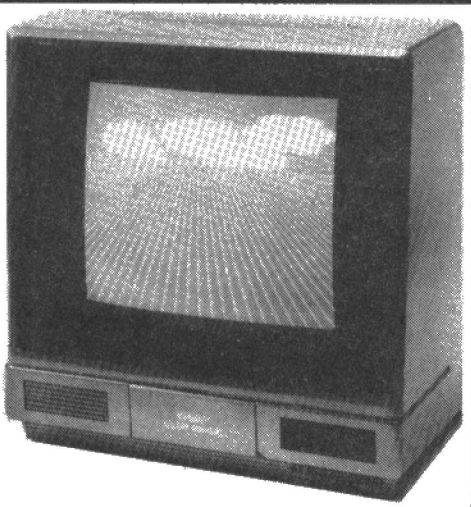
Gemini Microcomputers Ltd., 18 Woodside Road, Amersham, Bucks.
Henry's, 404-406 Edgware Road, London W2 1ED.

Software File

Software File and New Soft Releases is a new publication from EMAP Business and Computer Publications aimed at micro and software dealers, retailers, manufacturers and publishers.

The magazine lists over 1000 software packages.

EMAP, 8 Herbal Hill, London EC1R 5BJ.



Sub-£200 Colour Monitor ▲

A 14" colour monitor at under £200 is Fidelity's first product for the computer market. The monitor utilises the company's expertise in the domestic TV market. It accepts either RGB or RGBY or composite video inputs, together with audio, via a 21 pin peritelevision socket. The CM14 will therefore interface with any computer or games machine capable of driving a monitor.

Image definition is said to be outstanding with a 90° 14" in-line high brightness colour tube, with a video band width of over 12MHz. A detachable anti-glare tinted glass front is also provided.

Fidelity plc, Victoria Road, London NW10 6ND.

Slimline Disc Drive For The BBC

BBC and Dragon owners may be tempted by an elegant slimline disc drive, specifically designed for these machines and imported from Japan by Cumana.

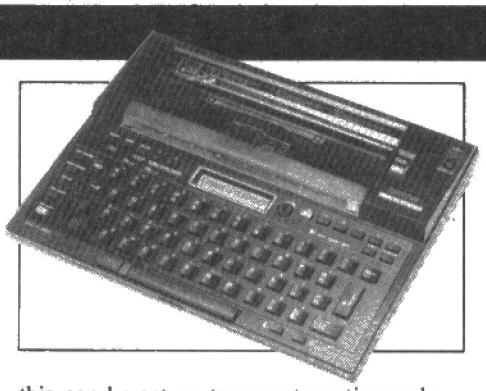
The 5¼" drives are available in single sided 40 and 80 track, and double sided 80 track formats. The design includes an independant power supply enabling up to two and four drives to be added to the BBC and Dragon micros respectively, without any modification to the computer or overheating.

Cumana Ltd., Pines Trading Estate, Broad Street, Guildford, Surrey. ▼

Typewriter and Printer in One

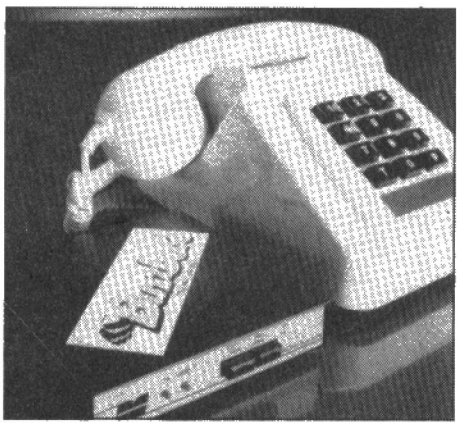
The perfect machine for the man too lazy to set up a wordprocessor file and a very useful and cheap device for any home computer owner is a combined printer/typewriter from Brother, the EP-22. This machine, complete with built-in RS232 interface, is eminently portable, in the real sense of the word – in that it is roughly the size of a small encyclopeadia and a good deal less heavy at 4lbs. The EP-22 is also eminently affordable with a price tag of £170 (inc. VAT).

The user has a choice of dot matrix thermal or ribbon printing. There is a comprehensive qwerty keyboard plus a 12 digit calculator. A 2K continuous memory will store about one A4 page of text and



this can be set up to repeat continuously. Correction facilities include a 32 character 'buffer' and a 16 character LCD screen with cursor controlled deletion, insertion and overtype.

Lowe Computers, Chesterfield Road, Bentley Bridge, Matlock, Derbys.



Buzzbox Modem ▲

DaCom Systems launched a new V21 modem at the Personal Computer World Show, the Buzzbox. Buzzbox is a full duplex modem (300bps) which is said to be far less prone to line errors caused by extraneous noise than the acoustically coupled modems which retail at a similar price. DaCom claim that the new instrument has a performance equal to that of modems costing three times its price of £80.

Buzzbox is battery-powered, with an optional mains adaptor for prolonged use. It has a standard RS232C interface, and has full British Telecom approval.

DaCom Systems Ltd., 200 Conniburrow Boulevard, Milton Keynes MK14 7AH.

Seven Colour Tandy Printer

The TRS-CGP-220 'drop-on-demand' ink jet printer from Tandy is a high quality (and high priced) machine capable of printing seven colours at 2600 dots per second, with a resolution of 640 dots per line. With parallel and serial interfaces (600/2400 baud) the printer is compatible with any TRS-80 computer. A screen printout utility will allow the printer to create multi-colour printouts of colour graphics screens produced from any graphics program. The price of the machine is £499.

68000 CPU Board

For around £1000 a complete 68000 development system, consisting of a Z80 host computer, cross assembler, 68000 CPU board and EPROM programmer can be obtained from Apollo software. The 16-bit microprocessor runs at 10MHz and contains 17 internal 32-bit registers and a 24-bit program counter. The on-board memory comprises fast EPROM and static RAM, allowing the processor to run at full speed without wait states. The standard board provides 16Kbytes EPROM and 4Kbytes RAM, expandable to 32Kbytes and 16Kbytes. A 24 line parallel interface and serial RS232C interface with a wide range of baud rates are also provided.

Apollo Software, Bucklebury Alley, Cold Ash, Newbury, Berks.

Joystick Interface

The Zedxtra intelligent joystick interface for the Spectrum was introduced at the ZX Microfair. The interface allows all software designed for use with the keyboard to be used with a standard 9-pin joystick (such as the Atari). It consists of a small box which plugs into the rear socket; an extension socket is provided for further expansion. A switch on the box turns on the interface, resulting in hardware reset to a program held in a CMOS RAM. The program puts a menu-selection display on the screen, and the user can use the keyboard to enter the names of up to sixteen games and the keys to be used for the five main joystick functions. The program calculates values for eight directions, with and without the fire button, loads all the data into the CMOS RAM, and resets to BASIC.

The interface can also be used as a pseudo-ROM: machine code routines can be loaded into the CMOS RAM, and made available above RAMtop when BASIC is running. This could be used to store a programmer's toolkit, or user-designated characters, for example. The unit is compatible with microdrives and printer, and has its own power regulator to avoid overloading the Spectrum regulator. The device is priced at £28.50 including VAT and p&p.

Zedxtra, 5 School Lane, Bournemouth, Dorset BH11 9DG.

Goodbye Breadboarding Blues?

Prototyping, repairing, or modifying PCBs using such methods as breadboarding or wire-wrapping will be, if Bishop Graphics have their way, a thing of the past. The Company has launched a PCB design kit (EZ8956) which uses adhesive backed conductive copper do'nut pads, component mounting configurations and tape which, it is said, do the same job much more efficiently.

The copper patterns and tape are simply applied to the board, and once the components and connections are soldered in the board is operational.

Each kit contains a complete assortment of adhesive backed conductive copper terminal strips, conductor strips, flat pack patterns, DIP patterns, TO patterns, do'nut pads and tape. Virtually everything needed to create single-sided, double-sided and multi-layered prototypes without artwork, photography, screening or etching.

Engineering and Electronic Design Services, Unit 7, Enterprise Centre, Childers Road, Limerick, Ireland.

Monochrome Monitors

Two high quality monochrome video monitors have been launched by Zenith Data Systems: they are the ZVM-122 (amber display), and ZVM-123 (green phosphor).

The monitors are compatible with the Commodore 64, Vic 20, TI-99/4A and Atari 800 and 1200, Apple II and IBM PC. They have a bandwidth of 15MHz, giving an ability to display more than 800 lines horizontally. Rise time is just 30 nanoseconds.

Both monitors are available at the price of £100 (plus VAT).

Zenith Data Systems, Bristol Road, Gloucester GL2 6EE.

Don't Damage Your Dragon!

Loading a cartridge or disc drive while, for example, a Dragon or Tandy Color Computer is switched on, is not a recommended practice and can result in serious damage. Human beings however, are fallible, and a range of new warning devices from Elkan Electronics may prove useful to the absent minded.

The 'Dragon's Eye' is an off-on indicator, priced at £3.95, which indicates when the computer has been on long enough to start overheating and also reminds the user that the machine should be switched off before adding cartridges etc. 'Dragon's Tail' is a joystick extender designed to make it easier to plug the joystick into the Dragon, and save wear and tear on the machine; it is priced at £2.95. The 'Dragon's Forktail' is a joystick Y-adaptor to enable the joystick port to be used for two purposes at once (price £3.95). All these products can also be used on the Tandy Colour Computer.

Elkan Electronics, 11 Bury New Road, Prestwich, Manchester M25.

For The Professionals ▲

Grandley titled a 'Superior Professional Competition Joystick' the Pro Ace from Sumlock Microware is designed as a replacement for all those battle fatigued Atari and Commodore joysticks which have reached their last legs.

The emphasis in this device is on strength. A high impact moulded case is used, with a steel shaft inserted into the plastic stick. The switches, which often prove to be the first part of a joystick to drop dead, have been bench tested to destruction by the manufacturers in over 1 million simulated operations.

The Pro Ace is fitted with a 9 pin D connector, suitable now for the Spectrum (with Interface 2) as well as the Atari and Commodore machines, and further connector types are planned for other home computers. The joystick has a two year warranty.

Sumlock, Royal London House, Deansgate, Manchester M3 3NE.

Atari XL Range

Two new Atari home computers and a range of printers and storage devices went on sale in October. The 600XL and 800XL are essentially updated versions of the Atari 400 and 800. The 600XL, with a recommended retail price of £159.99, has 16K of RAM expandable to 64K. Technical specifications include full-stroke design keyboard, Help key, software compatability with all Atari software, 11 graphics modes and four independent sound voices.

The computers are to be complemented by a range of low-cost printers and two new storage devices: a program recorder, the 1010, and the 1050 disk drive. The drive is priced at £300 and offers dual-density format. The printers vary in price from £200 to £400. As an example the mid-range 1027 is a letter quality printer said to be ideal for use in a word processor system such as the AtariWriter.

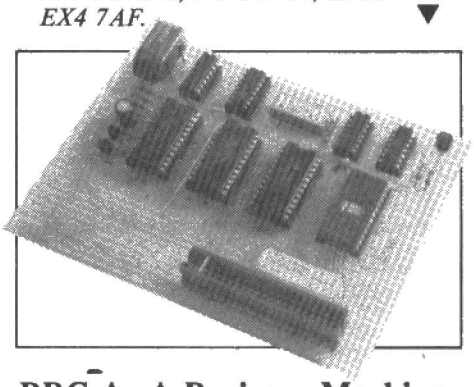
Ground Control, Alfreada Avenue, Hullbridge, Essex SS5 6LT.

ZX-81 RAM Board

The Micro-Z MZ-8 can store up to 10 programs in BASIC and/or Machine Code (a total of 6K), or provide 6K of extra memory for the ZX-81.

Three 6116 CMOS static RAMs are used to provide the memory storage; the unit is powered by an on-board NiCad battery, trickle charged from the computer when in use; an 8-way DIL switch controls memory decoding for the required use, and also activates the 'Write Protect' function. The latter is a very useful tool for the development of Machine Code programs: after a program has been saved, the 'Write Protect' switch is activated so that even if the program crashes or the MZ-8 is disconnected the programs will not be lost. Another feature of this device is a screen display of system status information.

Micro-Z Ltd., PO Box 83, Exeter EX4 7AF. ▼



BBC As A Business Machine

Acorn will soon be marketing Compact Software's system generator called Nucleus, to be sold in a software package with the BBC Micro Z80 add-on processor, running on CP/M.

The system is said to enable a relatively inexperienced user to develop business systems quickly and easily. The micro package is aimed at the larger personal micro user, and the smaller business user.

In addition, Compact are developing a Day Book package for Acorn – a system designed by accountants to meet the specific needs of a small business.

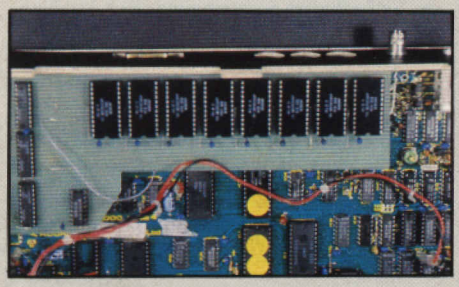
This package is an entry level system for the user requiring a business application, but not necessarily a fully integrated accounting system. The Day Book system would, for example, run in a very similar way to that in which a retail business would be accustomed to processing cash receipts. The user would have immediate access to simple management reports, and additionally the software can easily be upgraded to a fully integrated system if required.

A Wizard for the Genie

The first disc interface for the Colour Genie has been designed by General Northern Microcomputers. The 'Wizard' incorporates both disc and parallel printer interfaces, and also a disc operating system called QDOS. The Wizard will cost £99.

Gnomic, 8 Whitworth Road, South West Industrial Estate, Peterlee, Co. Durham.

To complement their EPROM programmer (see *E&CM* October, November 1983) Peter Simpson and Brian Alderwick describe a Sideways RAM Board project which does away with the costly and time consuming process of debugging programs written straight onto EPROM. Substituting a RAM board enables the programmer to run and test the software before committing it to read only memory.



SIDEWAYS RAM BOARD FOR THE BBC MICRO

When developing software to program into an EPROM, it is an optimistic person indeed who expects it to work first time. If it does not, the EPROM must be erased with ultraviolet light and reprogrammed with corrected software. Sideways ROM software development would be much quickened if the erasing and reprogramming steps were removed. Random Access Memory (RAM) has the ability to be directly programmed by the microprocessor and changes can be made very easily. This article describes a RAM board which can be substituted for a sideways ROM and which will allow programs up to the full 16K to be developed and run in situ. Once debugged, the final program can be saved to disc or cassette for subsequent loading into an EPROM.

The RAM sits in the same part of the memory map as any other sideways ROM i.e. &8000 to &BFFF; the machine operating system (MOS) does not differentiate between them. A description of the general working of the sideways ROM sockets was given in the article mentioned above and will not be repeated here.

Whilst developing this board, it became clear that it would prove useful to people who had bought more sideways ROMs than could be fitted directly into the BBC micro. The normal solutions to this problem would be either to buy an expansion board or to plug in the ROMs only as they were needed. However, this project offers a third alternative, namely, to save the contents of the excess ROMs to cassette or disc whence they could be loaded into the RAM board at will. Thus the four available sockets could be filled with a BASIC ROM, a Filing System ROM, the RAM board and your other most used ROM, say a word processor. Any other ROM programs could then be loaded into the RAM card as required. This alternative philosophy may appeal to those not wishing to build the EPROM Programmer.

The complete hardware for this project is described, together with a program listing which will initialise the board and provide routines to LOAD and SAVE files to and from the RAM card.

Circuit Description

A full circuit diagram is shown in **Fig. 1.** The 28 connections from one of the sideways ROM sockets are extended by ribbon cable to the remote board. All necessary lines are buffered by ICs 1-3, which are 74LS245 bi-directional buffers. The address lines and certain control signals which are uni-directional pass through IC1 and 2, which have the direction pin 1 tied to 0 volts. Buffer IC3 handles the data which, of course, needs to be bi-directional. In this case pin 1 is controlled by the read/write signal which is brought up separately, by a flying lead, from the main board.

All buffered outputs, except A11-A13 and chip select (CS), together with the read/write signal are passed to the eight RAM chips. These 21 lines are wired in parallel and are common to all RAMs.

The address lines A11-A13 are used to decode which of the eight RAM chips will be operational at any one time. This is done in IC4, a 74LS138, which is, as might be expected, a three to eight line decoder. The CS line enables IC4, which in turn allows any one of the eight RAM chips to be enabled as selected by the decoding. Several different makes of RAM chip were used in the prototype but 6116A-3 types with a 2K x 8 bit architecture are recommended since they are cheap and readily obtained. There are several alternative RAMs available in this style which should be equally suitable. All these ICs are of the 'static' RAM type because they offer reductions in circuit complexity and power consumption, however, they do cost somewhat more. To keep the heat dissipation inside the micro to a minimum, it would be best to choose CMOS types such as the 6116 recommended above since these consume very little power on standby.

A simplified RAM board using only two 8K x 8 bit chips was considered, but rejected because of the high cost of these ICs.

If the constructor wishes to use less than the full 16K available on the card, ICs should be plugged in starting at position IC5 in ascending order to IC12. Thus any amount may be used from 2K to 16K in blocks of 2K. The software will check the amount of RAM on initialisation.

RAM Card Installation

The board is connected to the micro by a 28 way ribbon cable which is soldered directly to the RAM board, the other end having a 28

pin insulation displacement DIL header plug, which can go into any of the four righthand ROM sockets IC52, IC88, IC100 or IC101. In fact, a plug of this size was difficult to obtain and the authors eventually used a cut down 40 pin version.

An additional connection needs to be made on the main board to the read/write line. This can be picked up in one of two places. If Econet is not fitted a wire can be neatly soldered into the hole for pin 12 of IC89. The alternative is to solder the wire to pin 1 of IC72, or if this is socketed, to insert a very small spade connector into the socket in addition to pin 1 of the IC.

Fig 2 shows the location of the RAM board and the route the ribbon cable should take. The fixing of the board inside the micro is left to the ingenuity of the constructor, since this depends on what materials are available and whether he or she is willing to drill the case.

A complete listing of all the parts necessary to complete this project is given below.

Using The RAM Card

The software provided with the RAM card enables two functions to be carried out. The first is loading a file from the current filing system into the RAM card and the second is saving the contents of the RAM card as a file to the current filing system. After the initialisation program has been run, two extra commands which perform these functions will be available to the user. These are '*LR prog' which loads the file 'prog' into the RAM card and '*SR prog' which saves the contents of the RAM card as a file named 'prog'. It is important to leave a space after *LR and *SR since these commands will not otherwise be recognised.

After the initialisation program has been run the two commands will become available and will remain, even when BREAK or CONTROL/BREAK is pressed, until either a *FX247,0 command is issued followed by a BREAK or the code itself is corrupted (possibly by injudicious poking).

The initialisation program has been tested with BASIC 1 and BASIC 2 together with the TAPE and DISC filing systems and runs on any combination of these without modification.

RAM Card Initialisation

The initialisation program should be the first program to be run after switching on the computer. When run, the program checks to see whether it is in the correct place in memory. If not then it moves itself automatically to the correct location and restarts itself. The program then examines every paged ROM slot to see whether it contains the RAM card. If the RAM card is not found then the message RAM CARD NOT FITTED is printed and the program ends. Assuming that the RAM card has been found then the program determines the amount of RAM fitted and prints a message saying how much RAM was found and in which slot. The program then asks where to code for lading and saving to the RAM card is to be located. Three options are provided. Firstly in the cassette buffer, secondly in the user defined character buffer and thirdly above the paged ROM workspace. The user simply has to select which of these is required and the program will assemble the code in the specified location and connect it to the operating system in such a way that the commands will still be directly available even after pushing CONTROL/BREAK.

The choice of location for the code will be influenced by the nature of the current filing system and by the features which subsequent programs will use. For example, the code should not be put in the cassette buffer if the cassette filing system is being used, nor in the user defined character buffer if characters are to be redefined. If neither of these options is open then select option three which puts the code on top of the paged ROM workspace (OSHWM). Normally PAGE is set to this value, however the initialisation code will ensure that PAGE is actually set to OSHWM +&100, thus protecting the extra code which is less than 256 bytes long. The only disadvantage of this is that PAGE will be 256 bytes higher than normal, i.e. &F00 on a cassette system or &1A00 on a disc system.

The initialisation program is quite long and will not fit into memory when modes 0 to 2 are used with a disc system, however, it may be shortened by leaving out REM's and assembler comments which are preceded by a backslash character '/'.

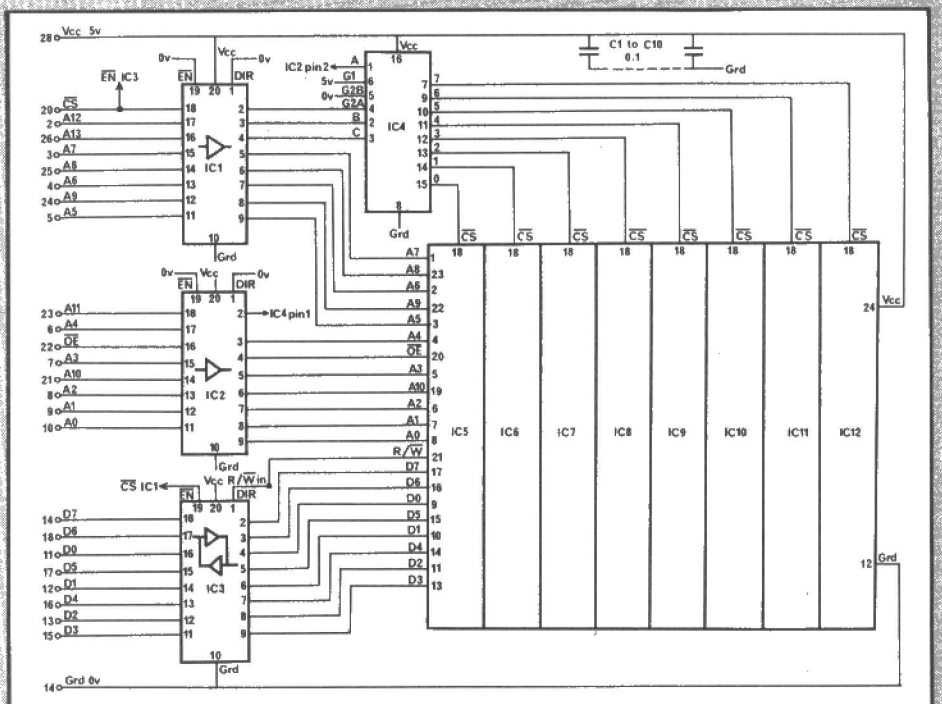


Figure 1. Full circuit diagram.

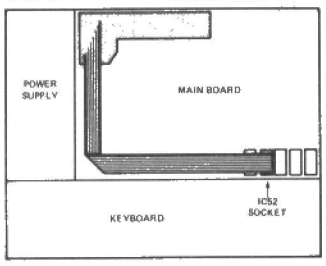


Figure 2. Location of RAM board.

PARTS LIST

TTL	IC1-3	74LS245	3 off
	IC4	74LS138	1 off
Memory	IC5-12	6116P-3	8 off
Capacitor	C1-10	0.1 microFarad disc ceramic	10 off
Sockets	For IC4	16 way DIL	1 off
	For IC1-3	20 way DIL	3 off
	For IC5-12	24 way DIL	8 off
Plug		28 way DIL I.D.C. header	1 off
Ribbon Cable		28 way by 450mm long (cut down 30 or 34 way)	1 off
Wire for links		Scrap resistor ends	
P.C.B.		90 by 240mm single sided	1 off

Hardware Details

The 12 ICs are mounted on a single sided printed circuit board. The foil layout is shown in **Fig 3** and the overlay in **Fig 4.** It is advisable to use sockets for all ICs, especially the RAM ICs 5-12, but the TTL chips can be soldered in as they are more robust. The tracks between the RAM locations are very fine and it is essential to use a fine point soldering iron. Note that pin 1 of IC4 faces north, all others face south.

There are ten capacitors on the board which act to de-couple the power supply lines. The sixteen wire links can be made from scrap resistor ends. Note that the link south of IC1 is bent into an 'L' shape.

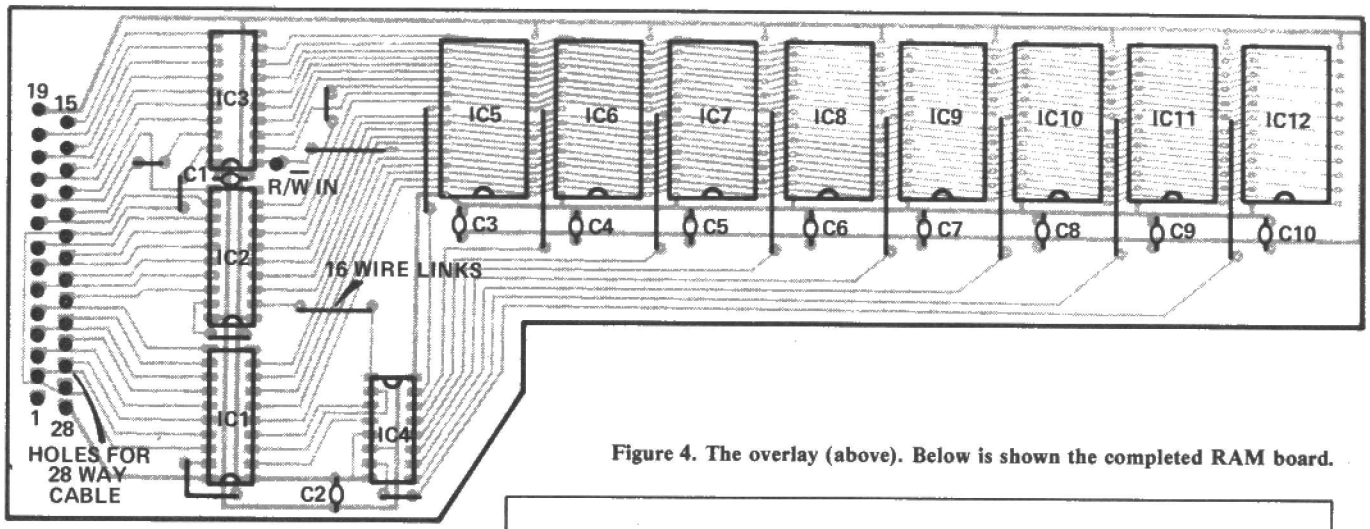


Figure 4. The overlay (above). Below is shown the completed RAM board.

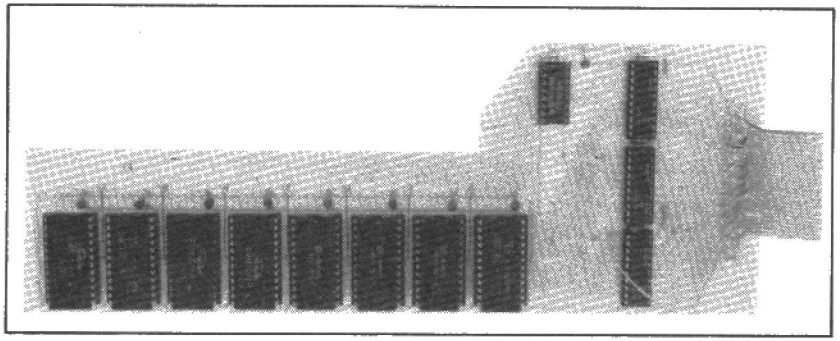
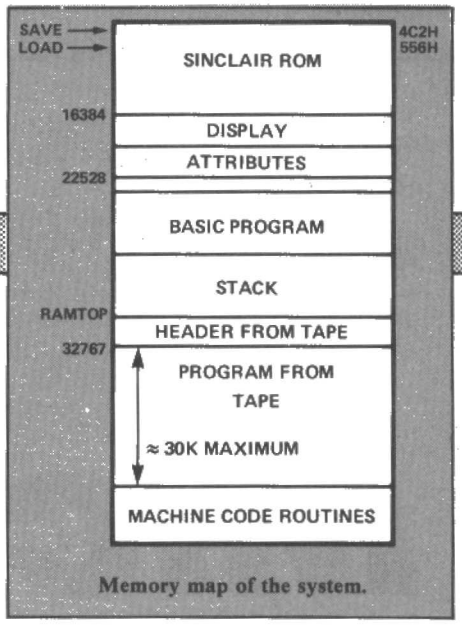


Figure 3. The foil layout.

48K SPECTRUM TAPE BACKUP



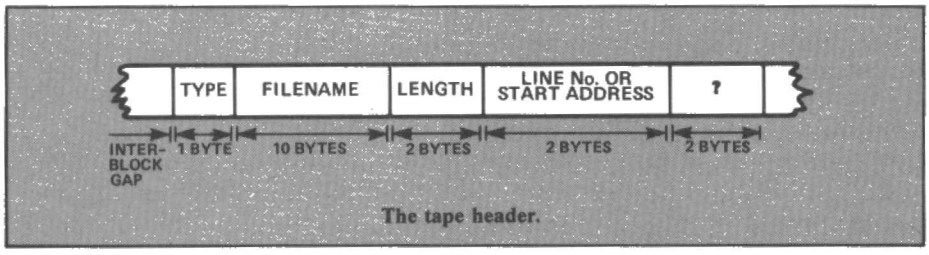
Memory map of the system.

Normally the 'LOADGO' system ensures that the only way to stop a tape between loading and running is to pull out the plug. This is not the most reliable means of achieving the desired end! The tape can be backed up into RAM by writing a separate load and save routine written in machine code.

The Tape Header

Each program that is saved on the Spectrum is preceeded by a header 17 bytes long:

TYPE:	This contains the type of the file 0 = Program is BASIC 1 = Numeric array 2 = String array 3 = Machine code
FILE NAME:	Contains the ASCII filename and is padded out with spaces if the filename is less than 10 characters long.
LENGTH:	Contains the length of the file in normal Z80 style i.e. LSB,MSB
START ADDRESS:	This contains the line number to auto-jump to or the start address.
LINE No.	As above



The tape header.

A short machine code routine was written to load in the header from tape into a particular memory position. The code follows using Z80 mnemonics:

```
AGAIN:  LD DE,17        ;Length of the header
        XOR A           ;Clear A
        SCF             ;Set the carry flag for loading
        LD IX,32768     ;Put header to memory at 32768
        CALL 0556H      ;Call the Spectrum loading routine
        JR NC,AGAIN     ;Try again if not header
        RET             ;Exit to basic
        CALL 0556H      ;Load it
        RET             ;Exit to BASIC
```

As soon as the program has been loaded the above routine will then return to BASIC. The program then utters a noise and asks the user to push any key when ready to save. At this point the tape to be written on is inserted and the plugs should be changed (MIC).

Saving On Tape

The program then calls yet another machine code routine which saves both the header and program from memory onto tape. Since the header is in memory it can simply be written out followed by the program.

The following code is the saving routine:

```
        LD IX,32768     ;Start of header
        LD DE,17        ;Number of bytes in header
        XOR A           ;Save header flag
        CAAL 04C2H      ;Save it on tape
        LD B,50         ;A delay loop between header
DELAY:  HALT            ;and program data
        DJNZ DELAY
        LD IX,0000      ;Start of program*
        LD DE,0000      ;Length of program*
        LD A,255        ;Save program
        JP 04C2H        ;Do it!
```

Within The ROM-Loading

The Spectrum ROM has a tape loading routine at 0556H. This routine will load the bytes from tape to a particular memory position. The input parameters to this routine are:

REGISTER A	0	=Load header
	−1	=Load program
CARRY FLAG	SET	=Load data
	RESET	=Verify data
REGISTER IX		=Memory to write to
REGISTER D		=This should be between 1-254 to be compatible with the ROM

The code used is very similar to the 'LOAD' command routine at 0761H except that it has a different value for IX.

The Program On Tape

When the program has displayed the decoded header it then proceeds to load the program (line 470). The machine code is similar to that of loading the header:

```
LD A,255            ;Loading program flag
LD D,19             ;D 0 or D 1
SCF                 ;Not verifying
LD IX,32768+40      ;Start address to load to
```

**These values are filled in by the program at lines 530-580. At the moment the start of the program is always 32768+40 but you can change this at lines 540,550 if you wish to convert this program for a 16K Spectrum*

Within The ROM-Saving

The Spectrum ROM has a tape saving routine at 04C2H. This routine will save code onto tape from a given memory address, the registers are:

REGISTER DE		=Length of code
REGISTER IX		=Start address of code
REGISTER A	0	=Save header
	−1	=Save code (Program)

The code used is similar to the code found at the 'SAVE' command at 0970H.

Once the program has been saved the computer emits a tone and asks the user if he/she wants to run again. This feature was included because programs are often saved in several parts. So to save all the parts you must repeatedly load and save until all the parts have been processed.

For example: The 'Flight Simulator' program (Psion) needs to be loaded and saved in four different parts:

1/FLIGHT	Length 39 BASIC
2/flight	Length 6651 BASIC
3/d	Length 6912 screen (m/c)
4/c	Length 25760 m/c

So in total the length is 39362 bytes (approx. 39K!).

Most commercial software packages run on a 'LOADGO' system which makes copying difficult. If, however, you want to read the machine code routines of that favourite space invaders program, or insure against a tape snapping, then this program from Tubb Research will overcome the problem.

Memory

Line 40 sets up the Spectrum as a 16K machine. So only approximately 30K at any one time can be loaded. This could be extended.

Graphics

You may have noticed the calls to a routine called GRAPHIC (at 65000). This routine causes the screen to fill with 'random' blocks of colour and then each colour disappears gradually in colour sequence. This makes for an interesting and eye-catching display. The routine to do this is only 57 bytes long and is shown below:

```
        LD HL,22528   ;Start of attributes
        LD BC,736     ;Number of bytes on screen
        LD DE,800     ;'Random' bytes from ROM!
BACK:   LD A,(DE)     ;Get 'random' byte
        LD (HL),A     ;Put in as an attribute
        INC HL        ;Screen position+1
        INC DE        ;Random number address+1
        DEC BC        ;Counter= counter-1
        LD A,B        ;Check if count
        OR C          ;equals zero
        JR NZ,BACK    ;If not do next block
        LD D,0        ;First control byte to look for
SEARCH: LD HL,22528   ;Start of attributes
        LD BC,736     ;Length
AGAIN:  LD A,(HL)     ;Get control byte from file
        RES 7,A       ;Remove 'flash' control bit
        CP D          ;the same?
        JR NZ,NO      ;If not then exit
        LD (HL),56    ;Put black ink, white paper
        PUSH BC       ;Now for a small delay
        LD B,0        ;B=Maximum count
WAIT:   PUSH HL       ;Delay factor
        POP HL
        DJNZ WAIT     ;Loop for 256 times
        POP BC        ;Restore count
NO:     INC HL        ;Attribute position
        DEC BC        ;Count-1
        LD A,B
        OR C          ;Is count equal to zero
        JR NZ,AGAIN
        LD A,D        ;Check for end
        CP 127
        REC NC
        INC D         ;Next control byte
        JR SEARCH     ;Search again
```

Each byte in the attribute file controls an 8*8 square on the screen of colour, brightness and flashing. They are held as binary as following:

BIT 7 FLASH ON/OFF
BIT 6 BRIGHT ON/OFF
BIT 5,4,3 PAPER COLOUR
BIT 2,1,0 INK COLOUR

So white paper, black ink, steady and BRIGHT=0 is:

BIN 00111000=56

(See Spectrum manual page 219).

Last Word

After using the program to backup tapes the system should be reset by using NEW. This is because of the routines within the ROM corrupt flags which stop the verify operation. So, programs which have been backed up using the program cannot be verified directly.

The program does not flag 'R-Tape errors' and simply returns to BASIC if there are any. So, if you have a bad tape with a corrupt ending you will still be able to back it up and then re-use it, retyping the end in.

For further details of the ROM, read 'The Complete Spectrum ROM Disassembly' by Dr Ian Logan, from the *E&CM* bookshop.

Decoder

The program could be modified to read in only the headers of the files and print these decoded onto a printer. So, if you had a tape full of files you could get a listing of all the filenames, lengths, etc.

Copyright

Before you back up any program make sure you are not violating any copyright notices. This program is for use purely as a utility for backing up programs and not for pirate copying.

```
TAPE BACKUP PROGRAM
By Tubb Research
  10 REM =TAPE BACK-UP PROGRAM=
  11 REM =For 48K Spectrums=
  20 REM =(C) Tubb Research=
  30 REM =Set UP RAMTOP=
  31 REM Maximum=32M program
  40 CLEAR 32767
  50 REM =Initialise variables=
  60 LET header=65000: LET save=
65200
  70 LET program=65295: LET grap
hic=65000
  80 REM Put in graphics routine
  90 LET byte=56: LET add=65000:
LET res=820
 100 GO SUB 710
 110 REM =Print title=
 120 CLS
 130 PRINT AT 0,6; INK 7;"TAPE B
ACK-UP PROGRAM"; AT 1,6; INK 7;"© Tubb
 Research 1983"
 140 REM =Randomize USR graphic
 150 REM Put in load routine's
 170 LET byte=29: LET add=65200:
LET res=550
 180 GO SUB 710
 190 REM Start of real program
 200 PRINT AT 4,6; FLASH 1;"STAR
T SOURCE TAPE": AT 5,1;"Loading the h
eader into memory"
 220 LET a$=""
 230 REM =Load header=
 240 RANDOMIZE USR header
 250 REM Decode header data
 251 REM =Get the start line=
 260 LET a=PEEK 65000 +
 261 REM =Get the code bytes=
 270 LET code=PEEK a
 280 LET a=a+1
 290 REM =Get file name=
 300 FOR j=a TO a+9
 310 LET a$=a$+CHR$ (PEEK j)
 320 NEXT j
 330 LET a=a
 331 REM =Get the length=
 340 LET length=PEEK a+10+256*PEEK
 (a+1)
 350 LET a=a+2
 351 REM =Get start address=
 360 LET start=PEEK a+10+256*PEEK (
a+1)
 370 REM Display decoded header
 380 PRINT AT 6,1;"HEADER:-"
 390 PRINT "FILENAME:";a$
 400 REM =Get file type=
 410 RESTORE 1000
 420 FOR i=0 TO code: READ t$: N
EXT i
 430 PRINT "FILE TYPE:";code;"-"
;t$
 440 PRINT "LENGTH:";length
 445 PRINT "START :";start
 450 PRINT AT 12,0;"LOADING ";t$
 451 PRINT "Please leave tape ru
nning"
 460 REM =Load program=
 470 RANDOMIZE USR program
 471 BEEP .3,12
 472 PRINT AT 12,0;"THE ";t$;" H
AS BEEN LOADED"
 480 PRINT "Ready for saving, pl
ease swap"
 485 PRINT "plugs and put in des
tination"
 490 PRINT "tape,push RECORD on
the tape recorder."
 500 REM Put in save's D/C
 510 SET file=27: LET add=65200
 LET res=550
 520 GO SUB 710
 530 REM Start address of progra
m
 540 POKE 65296,49
 550 POKE 65295,128
 560 REM Length of program
 565 LET a=INT (length/256)
 570 POKE 65301,length-(256*a)
 580 POKE 65302,a
 590 PRINT "Push any key when r
eady"
 600 PRINT AT 4,4; FLASH 1; INVE
RSE 1;"DESTINATION TAPE"
 610 PAUSE 4e4
 611 REM =Save the program=
 612 PRINT AT 21,0; SAVING ";t$
 615 RANDOMIZE USR save
 616 REM =Clear some VDU space=
 620 FOR g=1 TO 10: PRINT AT 11,
0,0
 621 NEXT g
 622 PRINT AT 4,4; INVERSE 1; FL
ASH 1;" STOP "
 623 PRINT AT 12,0;"Would you li
ke to run again?"
 630 PAUSE 4e4
 640 LET b$=INKEY$
 641 REM =Check for run again=
 645 IF b$="Y" THEN GO TO 120
 650 PRINT "Remember to type NE
W before you"
 651 PRINT "load in another prog
ram"
 652 STOP
 700 REM D/C to memory
 710 RESTORE res
 720 FOR i=0 TO byte
 730 READ a: POKE (i+add),a
 740 NEXT i
 750 RETURN
 800 REM =M/C ROUTINES=
 810 REM Graphics Routine
 820 DATA 33,0,88,1,192,2,17,32,
0,25,119
 830 DATA 35,19,11,120,177,32,24
 840 DATA 0,33,0,88,1,192,2,126,203,1
 850 DATA 70,35,10,54,56,197,6,0,229,2
 860 DATA 10,54,56,197,6,0,229,2
 870 DATA 193,55,11,120,177,32,2
 880 DATA 127,205,86,24,221,24,2
 890 REM Load header routine
 900 DATA 17,17,0,175,55,221,55,
 205,86,5
 910 DATA 5,40,248,201
 920 REM Load program routine
 930 DATA 62,255,55,205,221,55
 48,126,205,86,5
 940 DATA 201
 950 REM =Save program routine=
 960 DATA 221,55,0,126,17,0,0,1
 75,205,194
 970 DATA 4,6,50,119,16,253,221,
 33
 980 DATA 0,0,17,0,0,62,255,195,
 194
 990 DATA 4
1000 REM File type lookup
1010 DATA "PROGRAM","NUMERIC ARR
AY
1020 DATA "STRING ARRAY","MACHIN
E CODE"
1030 REM =END OF PROGRAM=
```

SAT 16

In conjuction with Satellite Services, we present the first in a series of articles describing a powerful 16 bit computer system.

The SAT-16 System has been designed with a wide range of users and applications in mind and provides a good starting point in the 16-bit field. In this month and subsequent articles we will be describing the design, construction, software and expansion capabilities available.

The basic system contains three main cards: A microprocessor card, which is a dual processor design based on the 68000 and 68701, a RAM/ROM card, plus a floppy disc controller. A basic system block diagram is given in **Fig 1.**

The subject of this month's article is the MPU card. The card is based on the 68000 microprocessor but has a unique dual processor link-up. It also incorporates on-board RAM, an EPROM which contains the monitor, plus a serial RS232 interface and parallel data port suitable for interfacing to printers and other parallel devices.

Intefacing to the main system bus is done to the G64 bus standard via tri-state buffers. All data transfers are asynchronous for off-card and on-card resources. Off-card peripherals are treated as synchronous devices. Control lines for direct memory access operations (DMA) are also provided. All existing G64-compatible cards from other sources can be used with this system.

The card can be separated into two major parts: The main 68000 processor which uses the on-board memory and interfaces to the system bus and the 68701 which takes care of all user interfacing and data transfers to and from the 68000. However, before explaining how each part works in detail, it may be interesting to take a closer look at the processors used – namely the 68000 and 68701.

The 68000 Processor

The 68000 was the first 16-bit microprocessor from MOTOROLA and although not software-compatible with the Motorola family of 8-bit processors, its instruction set and internal architecture provides exceptional processing power while at the same time being relatively easy to understand. The 68000 is manufactured using N-channel HMOS technology which gives high density and speed and low power consumption. The device also only requires a single +5V supply. Separate pins are provided for each data line and address output which means that in its basic form a total of 16M bytes of memory can be addressed directly without resorting to multiplexing or paging.

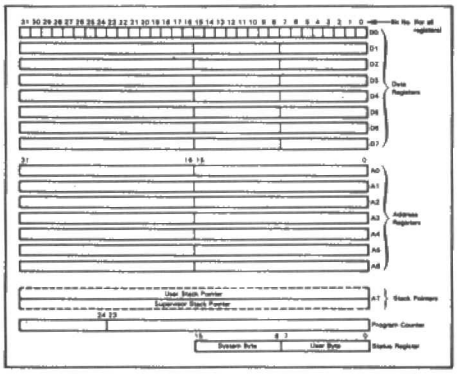


Figure 2a. Detail of the registers provided by the 68000 processor.

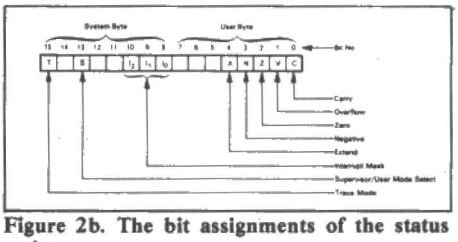


Figure 2b. The bit assignments of the status register.

Architectural Features

Figure 2a illustrates the registers provided by the 68000. There are seventeen 32-bit data and address registers, a 32-bit program counter and a 16-bit status register.

The data registers can be used to handle 8-bit bytes, 16-bit words or 32-bit long words. The 7 general-purpose address registers (A0-A6) can also handle either 16-bit words or 32-bit long word operands.

The processor can be operated in two modes – a supervisor (or system) mode, or in a user (or normal) mode. Separate stack pointers are used in each mode. Both pointers are addressed as Address Register A7. The status register is divided into two 8-bit bytes: The system byte and the user byte. **Fig 2b** shows the bit assignments for the status register.

The interrupt structure is arranged as 256 vectored levels. These are configured as 7 external interrupt requests, 57 internal system interrupts and 192 user vectors.

Provision is made to allow existing 6800 peripheral devices to be interfaced to the processor. Synchronous operation on the bus is also possible so that 8-bit system compatibility can be maintained, thus giving the user the ability to increase his processing power without having to resort to special 16-bit peripheral controllers.

Figure 3 illustrates the signals and pin

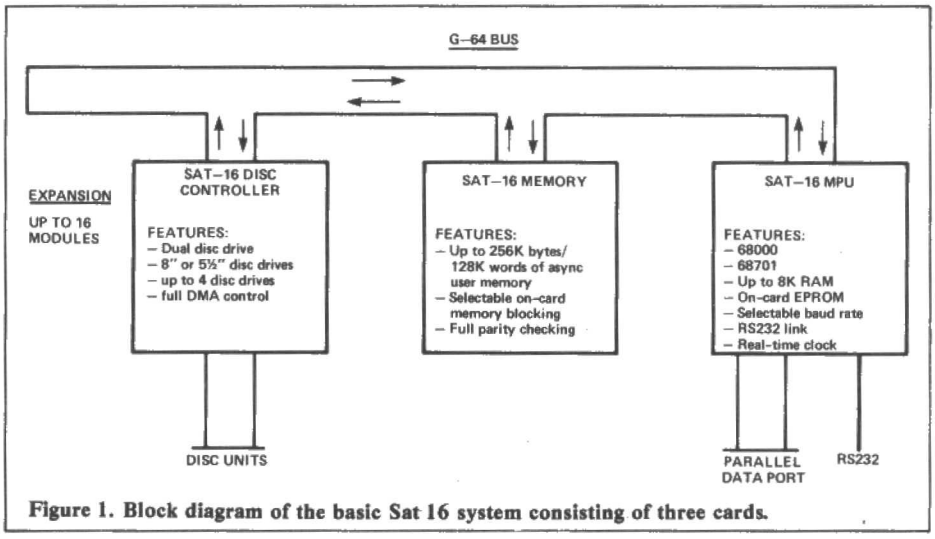


Figure 1. Block diagram of the basic Sat 16 system consisting of three cards.

FEATURES

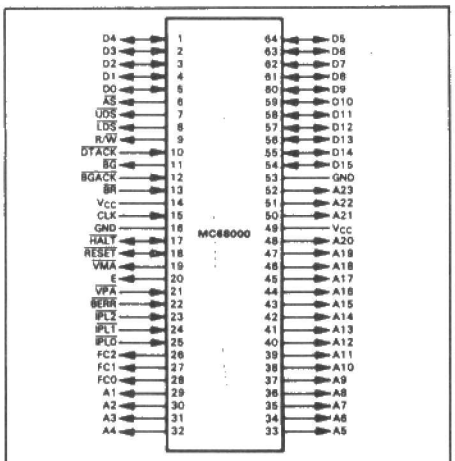
- ★ **Dual processor design 68000/68701**
- ★ **On-card EPROM and RAM up to 16K each**
- ★ **Programmable RS232 interface with four selectable baud rates**
- ★ **Programmable data port**
- ★ **Full DMA compatibility**
- ★ **User-accessible real-time clock**
- ★ **Easily expandable**
- ★ **On-card diagnostics**
- ★ **Dual processor capabilities**

assignments D0 - D15 are the bi-directional 16-bit data bus A1 - A23, are the output 24-bit address bus. Address A0 is not outputted to the bus but is used internally for size specification of each bus transfer, generating the UDS and LDS signals.

The UDS (Upper Data Strobe) and LDS (Lower Data Strobe) signals determine whether data is being transferred on either the upper (most significant) byte or the lower (least significant) byte or both bytes of the 16-bit data bus. DTACK is the Data Transfer Acknowledge input signal. This signal must be asserted by external logic during every asynchronous read or write cycle. All asynchronous devices in the system must include sufficient logic to general the DTACK signal. Function code outputs FC0, FC1 and FC2 are the function code or processor cycle status outputs. These outputs identify the type of bus activity currently being performed by the processor. Five different types of cycle are currently defined. These are accesses to either supervisor data memory, supervisor program memory, user data memory or user program memory and interrupt acknowledge cycle. Interrupt requests IPL0, IPL1 and IPL2 are the external interrupt request inputs.

These three inputs are decoded internally by the processor and compared with the interrupt mask to determine whether the interrupt request will be permitted. When all three interrupt inputs are low, a non-maskable interrupt (level 7, which is the highest priority), is present. When all three are high it indicates no interrupt is being requested.

BER is the Bus Error input. When this signal is low the 68000 performs a sequence (exception processing sequence) similar to that which it executes in response to an interrupt request. The purpose of BERR signal is to inform the processor when an external device has not responded (using the DTACK input) within an expected amount of time during a read or write operation.



Pin Name	Description	Type
D0-D15	Data Bus	Bidirectional, Tristate
A1-A23	Address Bus	Output, Tristate
AS	Address Strobe	Output, Tristate
R/W	Read/Write Control	Output, Tristate
UDS, LDS	Upper, Lower Data Strobes	Output, Tristate
DTACK	Data Transfer Acknowledge	Input
FC0, FC1, FC2	Function Code (status) Outputs	Output, Tristate
IPL0, IPL1, IPL2	Interrupt Requests	Input
BERR	Bus Error	Input
HALT	Halt Processor Operation	Input/Output
RESET	Reset Processor or Reset External Devices	Input/Output
CLK	System Clock	Input
BR	Bus Request	Input
BG	Bus Grant	Output
BGACK	Bus Grant Acknowledge	Input
E	Enable (Clock) Output	Output
VMA	Valid Memory Address	Output, Tristate
VPA	Valid Peripheral Address	Input
Vcc, GND	Power (+5 V) and Ground	

Figure 3. Pin-out and signal assignments of the 68000 processor.

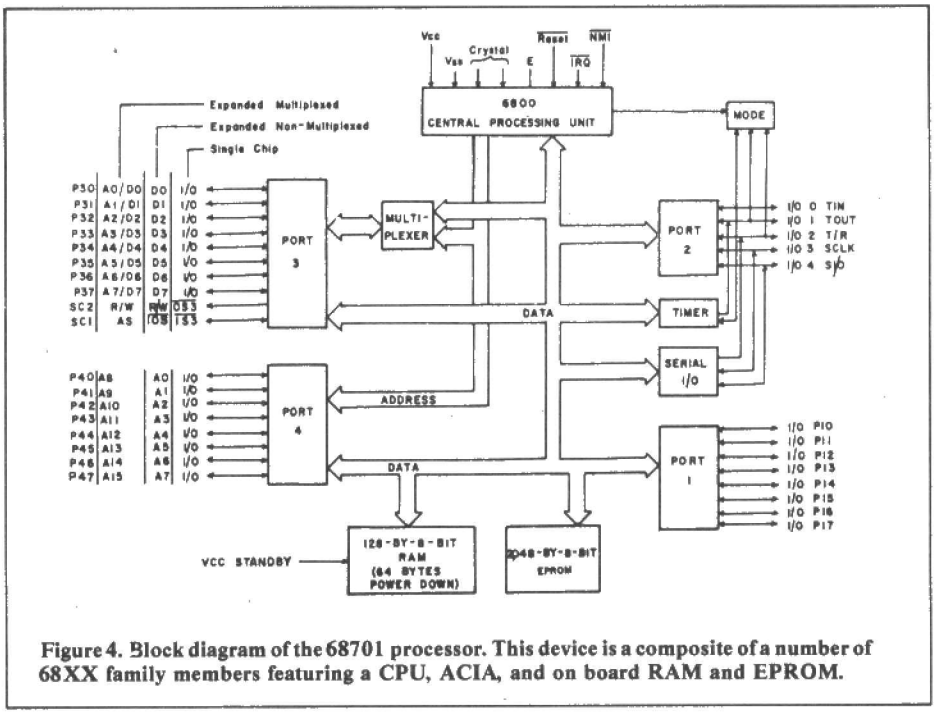


Figure 4. Block diagram of the 68701 processor. This device is a composite of a number of 68XX family members featuring a CPU, ACIA, and on board RAM and EPROM.

HALT is a bi-directional signal. If the HALT signal is asserted in conjunction with the BERR signal, the processor will automatically re-try the bus cycle that produced the error, when used alone it places the processor in a 'halt' state where the processor is essentially inactive until the HALT signal is negated. The HALT signal is also used in conjunction with the RESET signal to initialise the processor. One unusual aspect of the RESET signal is the fact that it is also an output signal.

HALT like RESET is also an output signal. If the processor detects two consecutive BERR conditions, HALT output is asserted. (This facility of the 68000 is used by the SAT-16 MPU Card to detect catastrophic failure of the system).

The processor provides a RESET instruction which, when executed, causes a low-going pulse to be outputted on the RESET pin, thus, you can execute a RESET instruction and use it to initialise other devices in the system without re-setting the processor.

CLK is the single-phase TTL-level clock from which all internal timing is derived.

BR (Bus Request), BG (Bus Grant) and BGACK (Bus Grant Acknowledge) are all bus arbitration signals. These signals are used to allow other devices such as DMA controllers and other processors use of the system bus. External devices request access to the system bus by asserting the BR input. The processor will then immediately assert BG and always relinquish the bus after it has completed its current bus cycle. The Bus Grant Acknowledge (BGACK) signal must be input to the processor by the device requesting the bus once that device takes control of the bus. The BGACK is essentially a "bus busy" signal that lets the other devices know that the bus is unavailable. The processor will only continue processing when the current bus controlling device negates BGACK.

The next three signals E, VPA and VMA are provided so that the processor can be easily interfaced to the standard 6800 family of peripheral devices. 6800-based devices use a synchronous method of effecting transfers of data. To accomplish this a system clock Enable (E) signal must be distributed to all 6800 devices so that all relevant data transfers may be synchronised to this clock. Thus the E provided by the 68000 is the equivalent of the 6800 (E) signal. The frequency of E is nominally equal to one-tenth that of the CLK.

The Valid Peripheral Address (VPA) signal is used by 6800-type devices to inform the 68000 that a synchronous data transfer is required. When the 68000 receives the VPA signal in response to a normal AS (address strobe), it alters the data transfer timing so that it is synchronous with the enable (E) signal. The 68000 will then assert VMA and E in the correct timing sequence to emulate a 68XX-type 8-bit processor. Synchronous 6800-type peripherals should use VMA as address validation not AS, and all data transfers will take place during E high time.

Addressing Modes

The 68000 utilises 14 different modes which can be grouped into six basic types, direct register addressing, direct memory addressing, indirect memory addressing, implied register addressing, program counter relative addressing and immediate data addressing. These different addressing modes create a powerful instruction set which will be the subject of another article. We can say at this point, however, that when compared to other microprocessors the instruction set is relatively easy to learn and very powerful.

The Other Processor

Why have a second processor just to handle I/O? Why not – it keeps chip count down and makes board layout easier, but that's not all, just imagine the flexibility. All the ports are effectively software re-configurable, so with the press of a key you can configure the I/O

ports to interface to all manner of devices. The device which allows such versatility is the 68701, a micro-computer chip from MOTOROLA which combines on one chip many functions which previously would have required large amounts of discrete logic. Besides 128 bytes of on-chip R/W memory and a 2K on-board EPROM, the 68701 also includes an on-chip timer and serial communication interface as well as up to 29 configurable I/O pins.

A block diagram of the 68701 chip structure is shown in **Fig 4.** The basic structure of the 68701 is actually a composite of a number of 68XX series devices. The CPU itself is an enhanced 6800 processor, the serial port is a 6850 ACIA, the on-board RAM is 6810 RAM device, the EPROM is a close equivalent of a 2716, and each port operates in a manner that is similar to the 6821 peripheral interface adaptor (PIA).

On-Chip Memory

The 68701 includes 128 bytes of on-chip R/W memory located at hex addresses 0080-00FF. A nice feature of the R/W memory is that the first 64 bytes (0800-00BF) can be retained in power-down or standby situations by using the standby function of the chip. Also 2K of EPROM is provided at addresses F800 through FFFF which can be programmed using a special programmer. This memory contains the specially developed software which controls the user interfacing on the SAT-16 MPU card.

I/O Ports

The extreme flexibility of the 68701 is provided by the various ways in which its I/O ports can be used. There are four ports – ports 1 to 4. The function of each port is determined by one of three modes, i.e. single-chip mode, expanded non-multiplexed mode, and expanded multiplexed mode. The expanded non-multiplexed mode is used on the SAT-16 MPU Card and will be described in detail later on. The various port functions are:

- **Port 1:** an 8-bit port which is always used as a parallel I/O port regardless of chip mode. It has an associated data direction register (DDR) and therefore each I/O line may be independently programmed for input or output.

- **Port 2:** A 5-bit port that can be used as a normal I/O port in all three modes. The port lines are configured as input or output with an associated data direction register. This port also provides access to the SCI and 16-bit timer. Therefore Port 2 can be five lines of parallel I/O, timer or any combination of these functions. It is also used to configure the chip for any one of its three modes of operation on power-up.

- **Port 3:** An 8-bit port which takes on different functions for the different chip modes. In the expanded non-multiplexed mode, Port 3 becomes the bi-directional data bus to provide data lines D0-07 for external operations. In the expanded non-multiplexed mode, these lines are mixed between D0-D7 and A0-A7.

- **Port 4:** Also an 8-bit port whose lines can be configured for parallel I/O or used as address lines.

Timer

The 68701 also includes an on-chip timer. The timer is effectively a 16-bit free-running counter. This can be programmed for one of four timing functions: Variable pulse width, continuous pulse width, single-shot pulse and period measurement. an input capture register is available; this register is used to store the present value of the count when an active transition is provided to P20. By reading the input capture register each time an active transition occurs, you can measure frequency or the time duration between external events.

Serial Communications

An on- chip SCI for serial I/O is also provided. It is capable of full and/or half-fuplex operation in standard mark/space format or bi-phase format for use between processors. the SCI also contains a programmable bit rate generator to provide four software selectable baud rates. It is also possible to configure the SCI to utilise an external 16x BR clock.

Clock

The crystal-controlled oscillator provided by the 68701 is used to drive both itself and the 68000 processor.

Card Description

System Memory

On-card memory is provided for use by the 68000 processor. Both RAM and EPROM is provided up to a total of 16K x 16-bit. The card will accept both 2732/2764 EPROM and 5564/6116 RAMS by a simple link change. The SAT-16 monitor (which will be the subject of another article) uses 4K bytes of on-card EPROM which leaves around 12K bytes of EPROM for user programs.

System Memory Map

The standard G64 specification allows for up to 128K words of memory but for larger multi-user tasks which may use up to 1 Mbyte plus, two extra address lines are available for use on an expanded version of the G64 bus and may be utilised on this expanded bus structure by fitting two optional links on the SAT-MPU Card. As mentioned earlier, the on-card memory is only accessible by the 68000. When accessing on-board addresses, the 68000 tri-states itself from the G64 bus; therefore overlapping on-board memory addresses may be used on the G64 bus as buffers for DMA devices which the 68000 will not see directly, i.e. no memory address contentions will occur for these addresses even though they are duplicated in the system. External memory addresses are considered as word address using A0-A18 on the bus to address 16-bit words whilst the upper and lower bytes of the word are selected by two data strobes DS0 (lower) and DS1 (upper).

Address A1 of the 68000 drives address A0 of the bus. 68000 address lines 20-22 are not used. A23 from the 68000 is decoded on the SAT-16 MPU as an asynchronous device address qualifier, i.e. a repeat of 1 MByte of peripheral addresses are available on the expanded G64 bus using VMA and E instead of AS. Addresses 0000 - 03FF are used for the system vector table, addresses 004000 - 0043FF are used as a copy of the vector table so that users can change vectors and thus handling routines to suit themselves.

Three interrupt lines are brought out to the G64 bus, (INT5, 6 & 7). INT7 is a non-maskable interrupt and will auto-vector to vector number 31. INT5 is maskable at level 5 and will auto-vector to vector number 29. INT6 is special; it is also maskable at level 6, but if asserted by the user, the processor expects the vector number to be provided by the interrupting hardware. RESET vectors the 68000 to start up software located at address 0400. The SAT-16 MPU memory map is shown in **Fig 5.**

Interfacing

All the cards within the SAT-16 System interface to the main bus which uses the G64 standard bus specification. This bus lends itself perfectly to control and industry application and can equally well be used for small specialised systems. The G64 bus is a 8/16 bus which has been designed to handle many common microprocessor systems working in a synchronous or asynchronous mode on euro-boards. Addressing capacity on the bus is not less than 256K bytes regardless of the I/O signals used. The connector pin-outs are given in **Fig 6.**

User Interfacing

As mentioned earlier, a second processor (68701) loaded with specially-developed software provides the serial and parallel interfaces for communication to the user VDU/printer and the main 68K processor as detailed below:

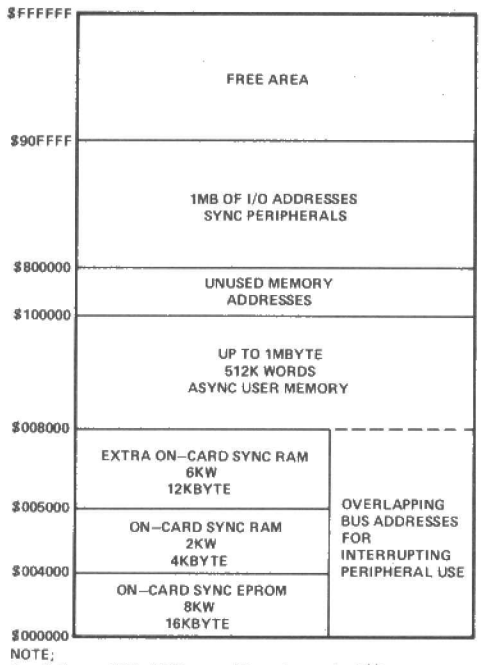
Serial Port (R232)

A full-duplex asynchronous serial communications interface (SC1) is provided with two data formats and a variety of rates. The SCI transmitter and receiver are functionally independent, but use the same dat format and bit rate. Serial data formats include standard mark/space (NRZ) and bi-phase and both provide one start bit, eight data bits, and one stop bit.

The start-up baud rate is selected by links which provide 300, 1200, 9600 or 76.8K options.

Parallel Port

Port 1 of the 68701 processor is used to interface to the user printer. The port is a mode-independent 8-bit port with each line an input or output as defined by the data direction register. Data is transferred as a 7-bit ascii character. Each character is strobed by the data strobe (PRSTRB line). Port 4 bit 4 is used for the 'printer ready' line (HI) to verify that the printer is available for data transfer.



NOTE;
i) Addresses 0000–03FF are used for system vector table
ii) Addresses 00400–0042FF are used for user vector table (vectors 64–255) when using INT6
iii) Main code starts at address 0400

Figure 5. Memory map of the SAT 16 system's memory.

ROW B		ROW A	
GND	1	GND	POWER(2)
A8	2	A0	ADDRESS (16)
A9	3	A1	
A10	4	A2	
A11	5	A3	
A12	6	A4	
A13	7	A5	
A14	8	A6	
A15	9	A7	
$\overline{BRQ}$	10	BGRT	CONTROL (18)
$\overline{RRQ}$ ($\overline{DS1}$)	11	$\overline{RGRT}$ ($\overline{DSO}$)	
$\overline{BGACK}$	12	$\overline{HALT}$	
Enable	13	MCLK (SYCLK)	
$\overline{RES}$	14	$\overline{VPA}$	
$\overline{NMI}$ ($\overline{INT7}$)	15	RDY ($\overline{DTACK}$)	
$\overline{IRQ}$ ($\overline{INT2}$)	16	$\overline{VMA}$ ($\overline{AS}$)	
$\overline{FIRQ}$ ($\overline{INT6}$)	17	R/$\overline{W}$	
IACK	18	Halt Ack	
$\overline{D12}$	19	$\overline{D8}$	DATA (8) High Byte
$\overline{D13}$	20	$\overline{D9}$	
$\overline{D14}$	21	$\overline{D10}$	
$\overline{D15}$	22	$\overline{D11}$	
$\overline{D4}$	23	$\overline{D0}$	DATA (8) Low Byte
$\overline{D5}$	24	$\overline{D1}$	
$\overline{D6}$	25	$\overline{D2}$	
$\overline{D7}$	26	$\overline{D3}$	
$\overline{Parity\ error}$			Miscellaneous (4)
$\overline{BERR}$	27	$\overline{Page}$	
Chain in	28	Chain out	
+5V Battery	29	–5V	Power (8)
–12V	30	+12V	
+5V	31	+5V	
GND	32	GND	

NOTE;
Terms in brackets are reserved for 16 bit or asynchronous systems.
Overlined terms are active or true when the voltage is low, other terms when the voltage is high.

Figure 6. The G64 bus specification.

Interface Adaptors

There are a number of different external I/O adaptors available for use with the SAT-16 MPU Card. The adaptor supplied for use with a serial RS232 VDU and Centronics printer is designed to work with the standard 68701 software. Adaptors for use with other external I/O devices are available with the associated 68701 software packages to drive them (e.g. local network adaptors, control software).

Note: It is not recommended to interface the SAT-16 MPU Card directly to any peripheral device without an adaptor.

Reset Operations

An external switch connected to pins 37, 39 and 40 of the connector J2 allow the user to re-initialise the SAT-16 MPU Card. A system reset line is also available on the G64 bus, but the processor will not perform a total reset unless both the RESET and HALT lines are asserted together for at least 10 clock cycles. Power on reset is automatically generated on the card.

Watch Dog

A watch-dog is not provided on the SAT-16 MPU Card but is available on the system-terminator card.

Real-Time Clock

The real-time clock on the 68701 processor is user-accessible via the RS232 link. The specially-developed software of the 68701 allows the user to start and stop the clock and set-up the initial value using specific user commands within the software.

CPU Clock

The clock for the 68000 and 68701 is generated by a crystal-controlled oscillator within the 68701 which gives a frequency of 4.9162MHz. This clock is also used to give the required baud rate settings on the 68701 serial port.

Specialised Circuits

Two programmable array logic chips are used on the SAT-16 MPU Card. These are the 16L8 and 20L10. These two arrays provide the address decoding for the on-card memory and the 68000 control signals to the G64 bus. These chips are configured and tested at the time the card is manufactured.

Processor I/O Controller

The AM2950 is used to control the parallel data transfers between the 68000 and the 68701 processors on an interrupt basis. INT3 tells the 68000 that data is ready from the 68701 and IRQ is used to the 68701 for the reverse direction.

Power Requirements

The power to the card is supplied by connector J1. The voltages shown in **Table 3** are required.

Software Facilities

Specially-developed software within the 68701 processor provides the user not only with the normal communication dialogue with the 68000 processor, but also a number of specialised user commands. The software is installed within the processor at manufacture. The software contained on the 68701 EPROM is by no means fixed. The user may utilise his own software to interface the SAT-16 MPU external I/O to other devices or systems. A special programmer is available from Satellite Services Ltd. for this purpose.

Special User Commands

CR, LF * / from the 68000, or */ from the VDU immediately following the prompt (?) tells the 68701 that the next byte is a command for itself.

The command sequence is not sent in either direction except for "*/?" command from the 68000 (see below):

C = toggle continuous mode
T = output corrent time to VDU
K = prompt user for new time
I = inhibit printer
P = restore printer
S = stop clock (if from VDU)
status request (if from 68000)
(** STA2 is sent to the 68000)
G = restart clock
R = reset clock
D = set direct mode (only from 68000)
Z = clear direct mode (only from 68000)
? = then send current time to 68000 (only from 68000). The time is returned to the 68000 as 6 bytes in the following format:
Byte 0 – *
1 – *
2 – hours (2 digit BCD)
3 – minutes (2 digit BCD)
4 – seconds (2 digit BCD)
5 – 1/100 secs (binary)

Note: An invalid command from the 68000 returns three "DEL" characters to the 68000.

Prompt to user VDU = ?

Self-Test

When the card is first switched on the 68701 software undertakes a self-test procedure on the internal RAM. During this sequence, the 68000 may also undertake a self-test routine. When both self-tests are completed, the 68701 reports the success or failure of these tests to the VDU.

Continued Next Month. E&CM

Micro Graphics Techniques

In part 3 of his series, Mike James looks at steering sprites and the creation of backgrounds and large objects.

In this part of Micro Graphics the subject of animation is brought to a conclusion with a short look at how users can be given control over the movements of the fundamental data type of animation – the sprite (see Micro Graphics Parts 1 and 2). Interactive control of sprites is such an obvious feature of nearly every computer game that it is often dismissed as a small problem of program implementation when in fact it is quite central to the success or otherwise of the game. The way that the user interacts with the game is important because it sets the level and type of skill required to be a successful player. A poor or inaccurate method of controlling the sprites simply raises the frustration level of the user because of the difficulty in getting the sprites to do what is required. Those who regularly play computer games cannot fail to have had the feeling that a key has been pressed or joystick moved in time to avoid destruction, but the sprite just didn't take any notice! On the other hand an accurate and natural method of controlling sprites produces a feeling of involvement that may raise a run of the mill game to new heights of popularity.

The methods that can be used to control sprites are to a certain extent governed by the hardware available. All computers have a keyboard of some type and so it is important to study ways that this can be used for control. Many computers also have a joystick input that is intended for sprite control but even this special purpose device has its problems and many interesting alternative ways of using it.

To bring the subject of animation to a close the subject of animating sprites against patterned backgrounds and animating large objects is considered.

General Sprite Control

So far the sprite animation loop – blank sprites, update sprites, plot sprites – has only been used to produce sprites that move around the screen bumping into things and occasionally exploding. However, interactive animation programs must allow the user a way of controlling at least one of the sprites in the display. This causes no real problem for the theoretical framework of sprite animation that has been constructed. The obvious place to insert any user control is in the 'update sprite' part of the loop. Also, the theory implies that whatever method of control is used it should be implemented as part of the force function – see Micro Graphics Parts 1 and 2. The simplicity of the sprite is partly due to the fact that its position and its velocity are automatically updated in the same way each time through the animation loop, and any variation in motion comes from the force function. It is also this fact that makes it easier to implement sprites in hardware – the regular updating can be carried out by hardware and the simple but more varied behaviour of the force function can be left to the software. As has already been demonstrated in earlier articles it is possible to exercise any sort of control over the motion of sprites using the force function, but it isn't always the most direct approach. In particular, user control over sprites is rarely directly in terms of a force function and it is usually more efficient, if a little less tidy, to alter the other quantities associated with sprites. User control can be classified according to which quantity is directly affected:

```
Position — direct updating of the x and y co-ordinates
Velocity — direct updating of the x and y velocities
Force    — direct updating of the force function
```

Each of these types of control actually occurs in games and other amination programs, each one has something different to offer and what is more surprising each one is encountered often enough to be considered as an important standard technique. For this reason each one will be dealt with in turn and considered both from the point of view of keyboard control and joystick control.

Positioning Sprites

Updating the x and y co-ordinates directly is the simplest form of control. When a keyboard is used as the input device every possible direction of movement is associated with a particular key. Each time through the animation loop the keyboard is inspected and, according to which key is pressed, the position of the controlled sprite is updated. For example:

```
10 X=10
20 Y=10
30 PRINT @X+32*Y," "
40 A$=INKEY$
50 IF A$="L" THEN X=X-1
60 IF A$="R" THEN X=X+1
70 IF A$="U" THEN Y=Y-1
80 IF A$="D" THEN Y=Y+1
90 PRINT @X+32*Y,"*"
100 GOTO 30
```

This simple program animates an asterisk. It is written in Dragon BASIC but it is not difficult to convert to other BASICs once you know that:

```
INKEY$          inspects the keyboard and returns the
                character corresponding to the key that is
                pressed. If no key is pressed then it returns
                the null string
and
PRINT @X+32*Y, positions the text cursor at X,Y
```

Although this program is very simple it does illustrate a number of important points of implementation. Firstly the keys U, D, L and R are used to control the movement of the asterisk simply because they are easy to remember as Up, Down, Left and Right. However, in a real time action game the physical positioning of the keys would be more important than any mnemonic quality. Most computers have a set of four arrow keys that are often used as direction control keys but it is often the case that the physical layout of the arrow keys would make another, less obvious, set of four keys more suitable. In particular the arrow keys on the ZX81 and the Spectrum are four adjacent keys on the top row and most professional moving action games do not use them for control. What makes a good set of control keys is something that I don't have a firm answer to. The keys should certainly be placed so that they don't get in each other's way. A favourite layout of my own is two directions per hand, eg up/down on two adjacent keys on one side of the keyboard and left/right on the other but a wide range of layouts seem acceptable. A second point illustrated by the above program is the use of an auto-repeat facility to increase the smoothness of control. On the Dragon the program produces one movement per keypress and so moving the asterisk around the screen is very tedious. On other machines, such as the Spectrum or the BBC Micro, the auto-repeat facility (or just the automatic clearing of the keyboard roll over buffer by INKEY$ or its equivalent) causes the asterisk to continue to move as long as the key is held down. Of course, there is a conflict between high repeat speeds making the sprite fast moving and difficult to position accurately and lower repeat speeds making the sprite sluggish but easy to position precisely. In general, if it is possible – as it is on both the BBC Micro and the Spectrum – the keyboard repeat should be adjusted to give the type of control that the application requires.

The use of joysticks as the input device for position control is in principle a lot easier than the use of the keyboard. All you have to do is read the x and y position of the joystick each time through the loop and then, after any scaling that is required, simply transfer the

joystick position into a screen position. However, in practice things are not quite so simple. For one thing the joystick's position can change a great deal in one animation cycle. In the extreme case this can take a sprite from one side of the screen to the other in one cycle! To see this try the following program:

```
10 GOSUB 1000
20 GOSUB 2000
30 GOSUB 3000
40 GOSUB 4000
50 GOTO 20

1000 CLS
1010 X=0
1020 Y=0
1030 RETURN

2000 PRINT @X+32*Y," ";
2010 RETURN

3000 X=INT(JOYSTK(0)/2)
3010 Y=INT(JOYSTK(1)/4)
3020 RETURN

4000 PRINT @X+40*Y,"*";
4010 RETURN
```

Once again this is written in Dragon BASIC but there should be no problems in converting it to other machines. Subroutine 1000 initialises the screen and X and Y. Subroutine 2000 blanks the current position of the asterisk and subroutine 4000 reprints it. The only subroutine that might prove difficult is 3000 which obtains the new readings for the position of the joystick scaled correctly for the size of the screen. In the Dragon's case the joysticks always return a number in the range 0 to 63 and this needs to be scaled to fit into a screen 32 columns wide and 16 lines high. If you run this program or its equivalent on your machine you will find that the movement of the asterisk is convincing until you move the joystick sharply from one side to the other – the asterisk appears to jump the entire width of the screen. This maybe what is required but in many games the fact that the asterisk can be 'pulled' around the screen in this way without passing through any of the intermediate positions can cause serious problems. A solution to this difficulty is to keep the sprite's position in one set of variables and the joystick's in another. At each update instead of simply setting the sprite's position to equal the current position of the joystick, the difference between the two is used to move the sprite closer to the joystick's position. For example, if you want the sprite to move by – at most – one position vertically and one position horizontally per animation cycle then change subroutine 3000 to:

```
3000 I=INT(JOYSTK(0)/2)
3010 J=INT(JOYSTK(0)/4)
3020 X=X+SGN(I-X)
3030 Y=Y+SGN(J-Y)
3040 RETURN
```

SGN(n) is a function that returns +1 if n is positive, −1 if n is negative and 0 if n is zero. Using this information you should be able to work out what the new version of subroutine 3000 does. The joystick's position is stored in I and J and the sprite's position is in X and Y. If the difference between I and X is negative, line 3020 subtracts one from X so reducing the difference. If the difference between I and X is positive line 3020 adds one to X, once again reducing the difference! Finally, if I and X are the same, line 3020 leaves well alone and adds zero to X! The same update mechanism applies to J and Y. This form of position control makes the asterisk appear to glide towards the current position of the joystick rather than jump erratically about the screen. You might think that this gliding would be unnatural or difficult to control but in fact in use it is very pleasing. If you don't require the sprite to move only one position at a time then there are a range of 'following' characteristics that you can use. For example, replacing lines 3020 and 3030 by:

```
3020 X=X+INT((I-X)/N)
3030 Y=Y+INT((J-Y)/N)
```

will make the sprite move toward the joystick, fast at first but slowing down as it gets closer to its final position. The value of the variable N governs how fast the sprite will move and so how closely it follows the joystick.

There are many other variations on positional control that can be used but they are all straightforward. The ideas introduced above should be used as starting points for your own experiments with positional control because it is very difficult to predict which forms of control are actually successful without trying them. One final point is that there is a lot of game potential in scrambling or coding the joystick inputs. For example, in the middle of a game it can add an element of excitement suddenly to discover that the way that the joystick affects the sprite has been reversed – ie moving the joystick up makes the sprite move down and vice versa!

Velocity Control

Velocity control is not used very much with joysticks because positional control seems to suit the situation much better. However, velocity control has a lot to offer if the only input device is a keyboard. Although this form of control is correctly known as velocity control, from the user's point of view it often seems more like directional control. The fundamental idea is that each time through the animation loop the keyboard is inspected and depending on which key is pressed, if any, the sprite's velocity is updated. In most cases a change in velocity brings about a change in the direction of motion rather than in the speed of the sprite – although a change in speed is sometimes useful. As an example of velocity control consider the animation of a 'bat' first using positional control:

```
10 CLS
20 X=15
30 Y=10
40 PRINT @X+32*Y,"   ";
50 A$=INKEY$
60 IF A$="L" THEN X=X-1
70 IF A$="R" THEN X=X+1
80 PRINT @X+32*Y,"XXX";
90 GOTO 40
```

where the XXX would normally be replaced by three solid block characters to represent the bat. This program moves the bat one position per keypress and as such is a little slow and tedious. Now try the velocity control approach:

```
10 CLS
20 X=15
30 Y=10
40 VX=1
50 PRINT @X+32*Y,"   ";
60 X=X+VX
70 IF X<0 THEN X=0
80 IF X>31 THEN X=31
90 A$=INKEY$
100 IF A$="R" THEN VX=1
110 IF A$="L" THEN VX=-1
120 PRINT @X+32*Y, "XXX"
130 GOTO 50
```

you will discover that the bat drifts either right or left depending on which key you press. Notice that the position of the bat is updated automatically each time through the animation loop even if no key is pressed and that it is the velocity VX which is changed in response to a keypress. This example is typical of velocity control in that things keep on moving even if you don't do anything. However, this simple one-dimensional (side to side) movement and control is not the only way that velocity control can be used. For example, as well as the obvious generalisation to two dimensions using fo - keys to control the up/down, left/right velocity of a sprite you can introduce a single or double key control that 'turns' the sprite. For example:

```
10 DATA 0,-1,1,0,0,1,-1,0
20 DIM V(4),W(4)
30 FOR K=1 TO 4
40 READ V(K),W(K)
50 NEXT K
60 CLS
70 X=15
80 Y=7
90 K=1
100 PRINT @X+32*Y," ";
110 X=X+V(K)
120 Y=Y+W(K)
```

```
130 IF X=0 OR X=31 THEN K=K+2
140 IF Y=0 OR Y=14 THEN K=K+2
150 A$=INKEY$
160 IF A$<>"" THEN K=K+1
170 IF K>4 THEN K=K-4
180 PRINT @X+32*Y,"*"
190 GOTO 100
```

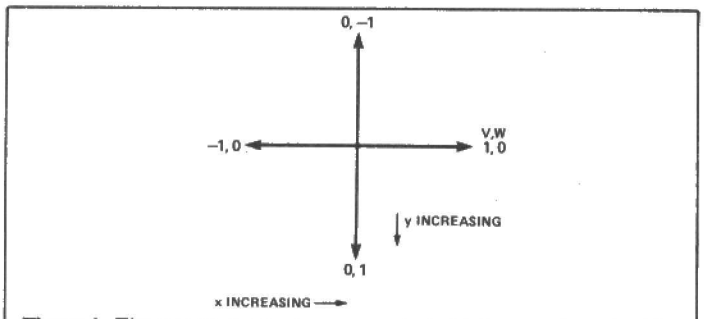


Figure 1. The program uses two arrays to hold the x and y velocities.

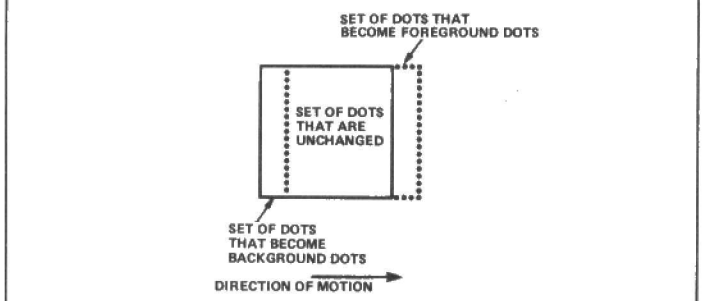


Figure 2. Illustration of how motion is obtained by redefinition of dots.

This program uses two arrays V and W to hoid pairs of x and y velocities as shown in **Fig 1.** The direction that the asterisk is travelling in is determined by the current value of K. Each time a key is pressed the value of K is incremented by 1 and a new pair of velocities is selected. This action makes the sprite turn through 90 degrees with each keypress. By adding four more elements to the arrays you can make a sprite turn through 45 degrees. This type of control is particularly effective if the shape of the sprite is changed so as to point in the direction of motion. The method of doing this is exactly the same as described last month for internal animation only instead of using the animation counter as an index to select the shape, the direction index (K in the above example) is used.

Velocity control could be used when a joystick is the input device and in this case the two components of velocity – direction and speed could be controlled. The x and y co-ordinates of the position of the joystick could be used, after appropriate scaling as the values of the sprite's x and y velocity. The only problem is adjusting the scaling so that the range of speeds available is sensible. For example, the previous joystick position control program can be modified to demonstrate velocity control by adding:

```
  35 GOSUB 5000
1010 X=15
1020 Y=7
3000 VX=(JOYSTK(0)-31)/16
3010 VY=(JOYSTK(1)-31)/16
3020 RETURN
5000 X=INT(X+VX)
5010 Y=INT(Y+VY)
5020 IF X<0 THEN X=31
5030 IF X>31 THEN X=0
5040 IF Y<0 THEN Y=14
5050 IF Y>14 THEN Y=0
5060 RETURN
```

Subroutine 3000 now updates the sprite's velocity according to the joystick's position. If the joystick is centered then the sprite will not move, otherwise it will move in the direction indicated by the joystick at a speed governed by how far away it is from the central position. This type of control is great fun to play with!

Acceleration Control

Acceleration control is not quite as common as the previous two methods but it does have one very important use. In games based on simulations of landing or flying spaceships, firing a rocket produces a change in the velocity of the sprite by applying a constant acceleration as long as a key is pressed. This sort of control is most simply implemented by directly updating the sprite's acceleration variables each time through the animation loop. For example, if a sprite is falling down the screen because of a constant vertical acceleration simulating gravity then a 'landing' game can be easily produced:

```
 10 CLS
 20 X=15
 30 Y=0
 40 VY=0
 50 AY=.1
 60 PRINT @X+32*Y," ";
 70 A$=INKEY$
 80 Y=INT(Y+VY)
 90 VY=VY+AY
100 IF A$="F" THEN AY=-.1 ELSE AY=.1
110 PRINT @X+32*Y," ";
120 GOTO 60
```

The 'rocket' engine can be 'fired' by pressing the F key and this will change the acceleration from .1 to −.1 so slowing the descent. All that is necessary to change this into a full 'lander' game is the addition of a fuel count and a sprite event detector for 'touch down'.

Acceleration control can be used in more complex ways, including two dimensional control and joystick control as with positional and velocity control. However, as these generalisations are not encountered very often and are very similar to the previous methods in implementation, they are not dealt with here.

Backgrounds

So far the least interesting subroutine in the sprite animation loop has been the 'blank sprites' subroutine. All this does is to print or plot a uniform block of background dots to remove a sprite from the screen. However, this only works if the background against which the sprites are moving is itself a uniform colour! In most cases sprites do not move across a uniform background. For example, in a space game the sprites would move against a background of stars or even planets. If you simply blank sprites by printing blanks then it is obvious that the background will slowly be eroded as the sprites move around the screen. Commercial video games machines often get round this difficulty by using printed transparent overlays that are stuck to the front of the screen. However, this is not a method that is suitable for home computer use. You might think that the simplest solution to the problem is somehow to save the area of the screen that the sprite is about to be printed at and then the next time through the animation loop the blanking subroutine could restore it. To use this simple scheme on most micros would require either some machine code or the use of PEEK to get the current contents of the screen. However, Dragon users have two commands ready made for the job: GET, which will read an area of the screen into an array; and PUT, which will restore the contents of the array to the screen. This method does indeed work but it doesn't solve the additional problem of what colour or pattern the sprite's own background dots should be. In most systems a sprite's shape is determined as a pattern of both foreground and background dots in a square array of dots. If the sprite's dots simply replace the dots on the screen then the patterned background will be destroyed even in places where it should show through. The solution is to avoid simply replacing the dots on the screen by the sprite's dots. Most micros allow you to combine screen and sprite dots using 'logical operation'. For example, ORing the dots together gives the following results:

screen	sprite	result
b	b	b
b	f	f
f	b	f
f	f	f

where b is a background dot and f is a foreground dot. If you examine this pattern you will see that the background dots in the sprite will not change the pattern of dots already on the screen – which is exactly what we need. The trouble is that in any given machine the way that logical operations are formed depends on the method used to generate

graphics. For example, the Spectrum's parallel attributes graphics OR dots together in the way suggested by the table but the BBC Micro's true colour display will only work in this way if you restrict yourself to two colours. The situation is even more confused because most micros offer a range of logical operations – OR, AND, NOT and EOR are typical. In general preserving backgrounds during sprite printing and blanking is not easy. To a certain extent this is because micro and BASIC designers have not yet realised the importance of sprites. Most machines do not have commands like the Dragon's GET and PUT, but there should always be a mode that allows sprites to be printed in such a way that only their foreground dots are transferred to the screen.

Animating Large Objects

Animating large objects is in fact not as common a requirement as you might imagine. This is fortunate because in general it isn't easy! The fact that sprites are small enough to make it possible to 'quote' their dot patterns makes it easy to build a simple theory and method of using them. Large objects generally take too long to draw to make it possible to use draw, blank and redraw animation loops possible in anything other than machine code and even then it is often difficult to make things happen fast enough. The best general method available is to make use of the observation that as a large object moves the dots that make it up fall into three groups:

- those that remain unaltered
- those that change to background dots
- those that change to foreground dots.

For most large objects the set of dots that are unaltered at each move account for most of the dots. This means that such an object can be animated by changing only the set of points that should become background dots and the set of dots that should become foreground dots. For example, consider the problem of animating a large square block so that it moves horizontally across the screen. As the block moves the trailing column of dots is changed to background dots and the leading column of dots is changed to foreground dots. The following program for the BBC Micro will animate quite a large block:

```
10 MODE 4
20 X=0
30 GOSUB 2000
40 X=X+1
50 GOTO 30
2000 GCOL 0,0
2010 MOVE X,500
2020 DRAW X,600
2030 GCOL 0,1
2040 MOVE X+100,500
2050 DRAW X+100,600
2060 RETURN
```

Subroutine 2000 will draw a line in background dots between X,500 and X,600 and then a line in foreground dots between X+100,500 and X+100,600. At first all you will see is a moving line but as soon as the line of background dots reaches the first line of foreground dots a moving square suddenly materialises!

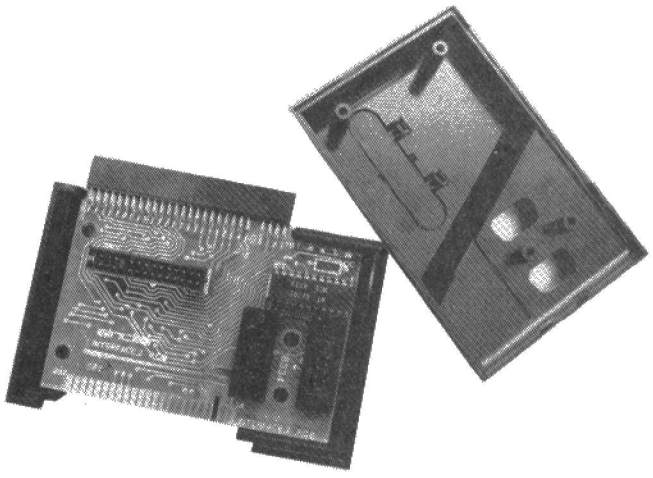
This method of only changing the dots that need to be changed sounds easy until you try it on a few apparently simple examples. Finding out the two sets of dots that change when a disk moves in a horizontal line is bad enough but if it moves along a curved path as in a simulation of a rising sun then things are really tricky!

Conclusion

There are many more topics concerned with special animation techniques that could be included in this discussion but the time has come to move on to two-dimensional static graphics. You will find, however, that many of the ideas are fairly obvious once you have understood the sprite method of animation. The most challenging areas of animated micro graphics at the moment are improving the methods of control and animating large objects – you might like to try to write a smooth 'sun rise' program using nothing but BASIC. If you succeed add a few clouds and I'm sure other *E&CM* readers would be interested!

Next Month – Static two-dimensional graphics.

THAT INTERFACE 2 REVIEW



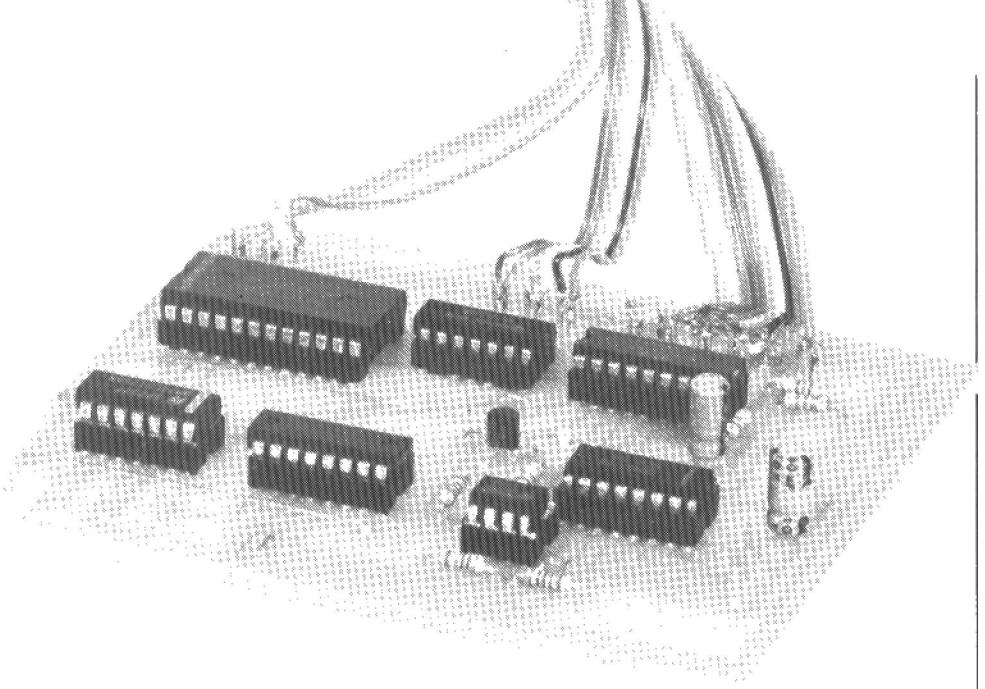
In last month's *E&CM* a full review was promised of the new Sinclair add-on, the Spectrum Interface 2. To renage on the assurance would be unforgiveable, but to be blunt, there is really little to be said about this device.

Take a 4 x 2 1/2" PCB, add a socket for the ROM cartridge, two 9-pin D plug sockets, and one solitary chip (the ubiquitous ULA). Cover this assembly with a matt black plastic case and shove on plug and socket fore and aft, and hey presto there you have it – Interface 2.

The device plugs directly into the Spectrum rear expansion port, or into the rear port of ZX Interface 1. No extra interface is needed to match the joystick to the Spectrum or software, and, once connected, the joystick will work with either ROM cartridge, cassette or Microdrive programs. A socket is available at the back of the Interface into which the ZX Printer plug can be inserted.

The games cartridges, as neatly styled as the Interface itself, are encapsulated in a package about half the size of a matchbox. They are inserted into a porthole at the top of the Interface, and immediately the computer is switched on the title piece of the game flashes onto the screen – no more hours of waiting for those cassette games to load, no more fiddling with the volume controls. The cartridge connector is enveloped by a rubber skirt which is designed to protect the connector from wear and corrosion. The cartridges fit very tightly into the Interface and extraction requires a swift yank; this is one potential problem area: the connection from Interface 2 to the computer has ominous similarities to that between the ZX-81 and its 16K RAM pack. It is quite likely that the continuous strain of insertion and extraction of games cartridges will result in a deterioration of the connection, and the device would best be used with a ribbon cable connection. A further omission from the design is a fail safe device to prevent insertion of the cartridges when the computer is switched on – an inevitable event at some stage which could have unfortunate results for the computer.

All in all Interface 2 is a very competitive product. Priced at £19.95 it will outperform the Microdrive and Interface 1 for the games only enthusiast (a sizeable minority). A good deal of heartache will be suffered by the software manufacturers faced with the choice of writing on cassette, looped tape or cartridge. Currently Psion are writing all the ROM games cartridges (of which ten are currently available) and this is a situation unlikely to change until the success of the product is assured. The cartridges cost nearly as much as the Interface itself, at £15 each, and three times as much as most cassette software.



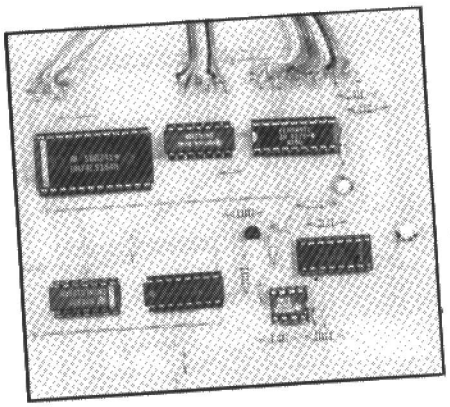
ELECTRON A/D CONVERTER

The Acorn Electron, equal to the BBC Micro in many respects, has only the most basic of built-in interfaces. An A/D converter by R. A. Penfold is the first of a series of projects which will give the Electron a potential approaching that of its big brother.

The user port, RS423 interface, analogue port and 1MHz bus which make the BBC microcomputer so versatile are all lacking on the Acorn Electron. An Electron add-on interface unit is due to become available in the not too distant future, but it is also possible to add user-built hardware direct onto the expansion bus at the rear of the machine.

A unit such as this forms the subject of this article. It is a four channel analogue to digital converter that also provides a couple of digital inputs. The unit could therefore be used with two joysticks and the digital inputs could be used for fire-buttons. The unit is also suitable for use in many measurement and control applications.

On the face of it this analogue port seems to offer similar facilities to the one fitted to the BBC machine, and presumably therefore, also the analogue port of Acorn's Electron add-on interface unit. However, there are a couple of important differences which should be borne in mind. One is that this unit is an 8 bit type whereas the BBC machine has a 12 bit converter. Although noise problems prevent the latter from achieving proper 12 bit resolution and accuracy, it probably offers slightly better accuracy than this 8 bit circuit. On the other hand, the BBC analogue port offers a maximum conversion rate of only about 100 per second, whereas this unit gives a maximum rate of around 65000 per second and is consequently somewhat more versatile. Although the circuit has only a single converter, a CMOS analogue multiplexer is used to provide four analogue inputs. This effectively reduces the maximum conversion rate if more than one channel is used, since each channel has to be converted in turn; but adequate speed can normally be obtained using this system, which is the one employed with most multi-input analogue ports (including the one on the BBC machine). The input voltage range is 0V to just under 5 volts incidentally.



Overhead view of the completed A/D converter board.

Edge Connector

The expansion bus at the rear of the Electron is actually the rear of the double-sided printed circuit board, and it is designed to take a 2 × 25 way 0.1 inch pitch edge connector. There is a slot for a (ZX81/Spectrum style) polarising key in the connector. The Electron manual does not provide any connection details, but **Figure 1** shows most of the edge connector functions. In this case only the address bus, data bus, supply lines, and read/write line are needed, all of which are available on the connector.

Only a very basic memory map is provided in the Electron manual, and this shows the input/output area as extending from &FC00 to &FEFF, which is the same as for the BBC machine. In the case of the latter page &FE is used for internal input/output devices while pages &FC and &FD are free for external add-ons. The Electron input/output arrangement would seem to be very similar, and probably identical. Pages &FC and &FD certainly seem to be free for user add-ons, and in this case page &FC is used. On the BBC machine the upper eight address lines are not available on the 1MHz Bus, and instead these are decoded to provide two outputs (one for page &FC and one for page &FD). This makes it possible to add one or two circuits to the 1MHz Bus without any extra address decoding, but, unfortunately, there would seem to be no equivalent to these outputs on the Electron's expansion bus. Any user add-on must therefore include suitable address decoding.

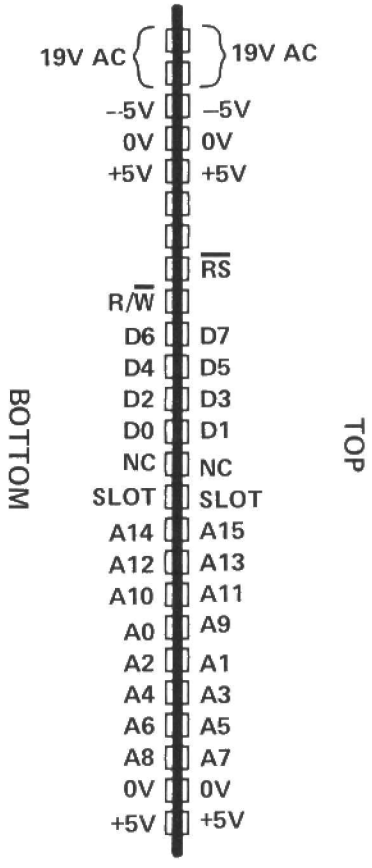


Figure 1. Edge connector functions of the Electron expansion bus.

Block Diagram

Fig. 2 shows the general arrangement of the analogue port in block diagram form.

The converter itself is a successive approximation type, and it has three state outputs which can be connected direct to the data bus. With this type of converter the input signal is compared with the output of a digital to analogue converter, and this has its inputs fed from the data outputs of the circuit. The outputs are initially set to zero, apart from the most significant bit which is set at 1. This gives approximately half the full scale voltage at the output of the D/A converter. If this voltage is greater than the input voltage the most significant bit is left at 1, otherwise it is set at 0.

Next bit 6 is set at 1, and again it is either left at 1 or reset to 0 depending on whether or not the output of the D/A converter is at a higher potential than the input signal. This process is repeated for bits 5 to 0, with the digital output gradually becoming more accurate until it represents a true reflection of the input voltage. This system provides a fast conversion time, and for an 8 bit converter it takes just nine clock cycles. The 2N427E device used in this circuit will operate with a maximum clock frequency of at least 600kHz and gives a guaranteed conversion time of as little as 1.5uS.

A "start conversion" pulse has to be supplied to the converter each time the input voltage is to be read, and this pulse is obtained from the address decoder. The address used is &FC00, and the value written to this is obviously irrelevant as the pulse is not obtained from a line of the data bus. The converter has a status output which goes high when a conversion has been completed, and remains in this state until a new conversion has commenced. This can be used to prevent the output data from being read until the conversion has been completed and the data is valid, but this facility is only really needed when using an assembly language routine. When using BASIC the operating speed is such that the time delay between sending the "start conversion" pulse and reading the output of the converter is long enough to ensure that conversion will have been completed.

A three state buffer is used to interface the end of conversion output to the data bus, and this additionally provides the two digital inputs. These buffers are enabled when reading address &FC01. The three state buffers at the output of the converter are enabled from the address decoder via an inverter stage, and reading &FC00 returns the value from the converter.

A four channel multiplexer is used ahead of the converter, and its two address lines are controlled from data lines D0 and D1 via a dual latch. The latching pulse is obtained from the address decoder, and writing (0, 1, 2, or 3) to &FC01 selects the desired channel.

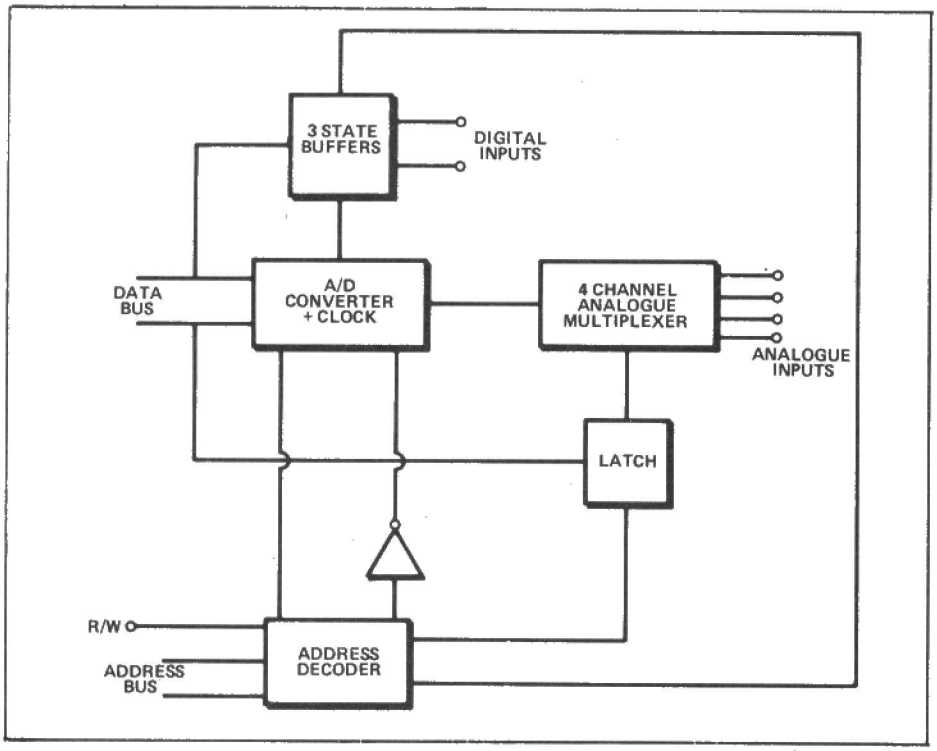
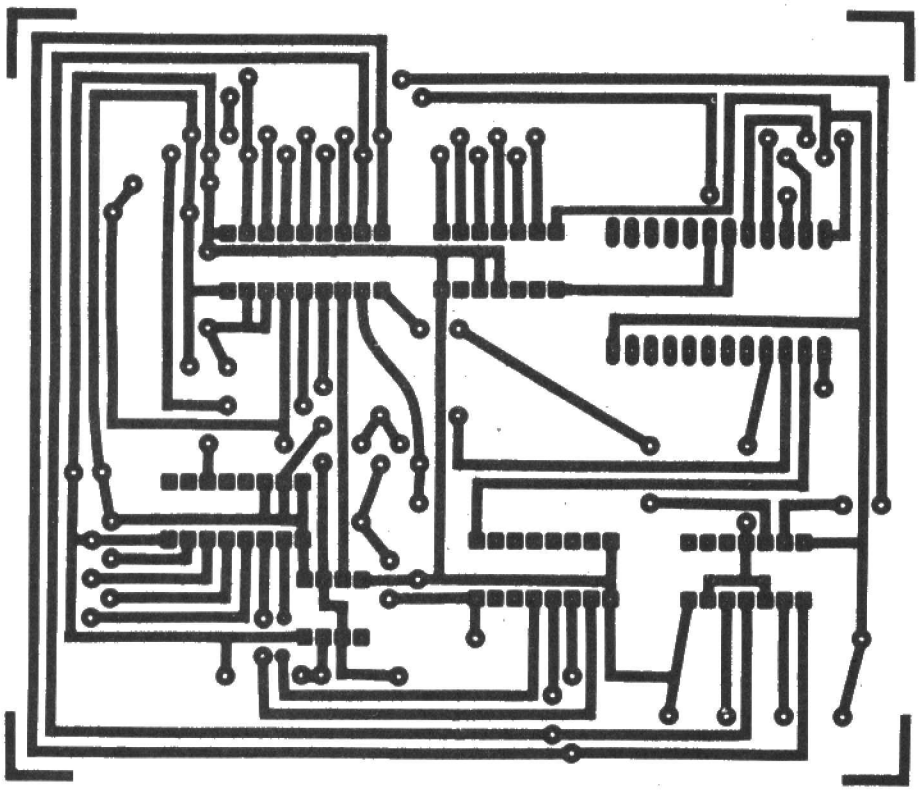


Figure 2. Arrangement of the analogue port.



PCB foil pattern, shown actual size.

The Circuit

Refer to **Fig 3** for the full circuit diagram of the analogue port.

IC3 is the converter, and R3 is the emitter resistor for the input stage of its high speed comparator; this must be fed from a negative supply to enable input voltages right down to the 0 volt supply to be compared, and the Electron provides a suitable −5 volt supply. R4 and C1 are the load resistor and decoupling capacitor for the internal voltage reference used in the digital to analogue converter. R7 and R8 form an attenuator which reduces the input sensitivity from 0 to 2.55 volts to a full scale value of just under 5 volts. These also provide a suitable source impedance so that good temperature stability is obtained.

The clock signal for IC3 is provided by a 555 astable based on IC7. The specified timing component values give an output frequency at something approaching the 600kHz guaranteed maximum for the ZN427E.

IC4 is the CMOS analogue multiplexer, and it is an eight input type, but in this circuit one address input is simply tied to the 0 volt rail and four of the inputs are ignored. The input selected depends on the binary number fed to pins 10 and 11. IC6 is a hex D type flip/flop, and two sections of this are used to form the latching outputs which drive the address inputs of IC4. Data lines D0 and D1 are used to control the address inputs of IC4 so that the number written to IC6 at &FC01

selects the analogue input of the same number.

IC5 is a quad three state buffer, but here only three sections are used. Two are used to interface the digital inputs to data lines D0 and D1 while the third interfaces the "end of conversion" status output of IC3 to D7.

Address decoding is provided by IC1 and IC2. IC1 is an 8 input NAND gate, but two inputs are effectively eliminated by being connected to the positive supply rail. The other six inputs decode address lines A10 to A15, and a negative output pulse is produced only when these are all high. This gives an enable pulse to the two negative enable inputs of IC2.

IC2 is a 4 to 16 line decoder, and it is used to decode A0, A8, A9, and the read write line. Four outputs are used. Writing to &FC00 gives a negative pulse from output 0 (pin 1) which activates the start conversion input of IC3. Reading from the same address gives a negative pulse from output 2 (pin 3). This is inverted by TR1 to give an enable pulse to the tristate buffers of IC3 so that the output of the converter is placed on the data bus and read by the computer. Writing to &FC01 provides a pulse from output 1 (pin 2) which latches the data sent to IC4. Reading from this address gives a negative enable pulse to IC5 so that the digital inputs and status output of IC3 can be read.

Note that the address bus is only partially decoded, and that the analogue port occupies all the addresses in page &FC (the two addresses mentioned above are probably the easiest to remember and use in practice). It is therefore not possible to use any addresses in this page for other purposes when the analogue port is in use.

There seems to be no problems if the unit is powered from the Electron's power supply, but if it is likely to be used for long periods of time it might be worthwhile using a separate power supply. The positive supply current is well under 100 milliamps (55 milliamps typical), and the negative supply current is negligible (about 65 microamps).

Construction

Details of the printed circuit board are provided in **Fig 4.**

Construction is fairly straightforward, but be careful to fit the integrated circuits with the correct orientation – these do not all fit onto the board the same way around. As IC4 is a CMOS device it should be fitted in a (16 pin DIL) IC socket, and it should not be plugged into circuit until the board is otherwise complete. Leave this device in its protective packaging until it is to be fitted in place, and handle it as little as possible. IC2 and IC3 are not the cheapest of components, and it is probably worthwhile using sockets for these even if the other integrated circuits are soldered direct to the board.

Veropins are used at the numerous points where the off-board connections are made. There are thirteen link wires, and construction will probably be easiest if these are left until last. About 22swg enamelled copper wire is suitable for the link wires.

Connection to the Electron is by way of a 23 way ribbon cable about 0.5 metres long, and a 2 × 25 way edge connector. It is unlikely that a 23 way cable will be available, but it is easy to cut down a 26 way or larger cable to the required size. Multicolour ribbon cable is preferable to the grey type as any crossed-over wires or similar mistakes will then be more easily spotted.

The printed circuit board has been designed so that the connections on the board are in the same order as those on the edge connector, and wiring the cable to the edge connector is not too difficult, However, make quite sure that the connections are made to the right terminals of the connector. Before connecting the cable to the connector, strip

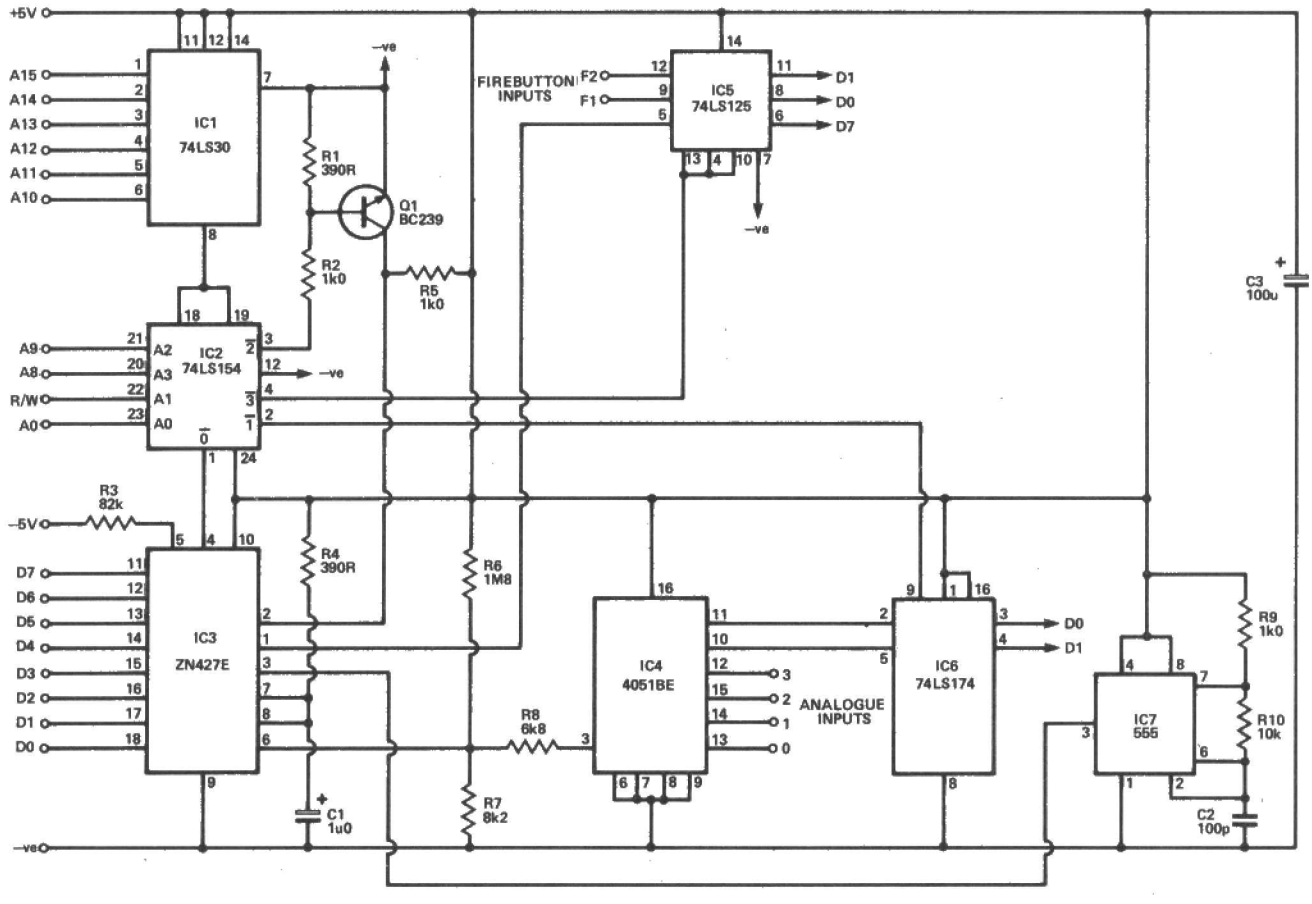
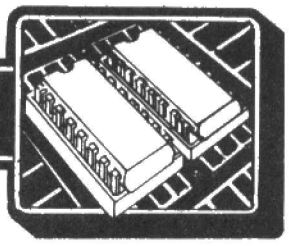
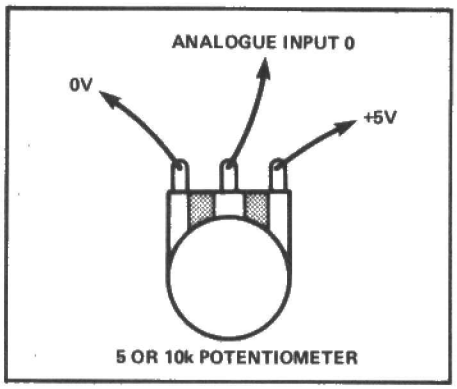


Figure 3. Full circuit diagram.



the insulation from the end of each lead, and then tin the bare ends plus the appropriate terminals of the connector with a small amount of solder. Note that **Fig. 1** shows the Electron expansion bus as viewed when looking onto the rear of the computer.

Before connecting the unit to the Electron carefully check all the interconnections at least once. It could be disastrous if the edge connector is fitted the wrong way round, and it would be worthwhile to try fitting a stout piece of wire into the connector at the appropriate place to act as a polarising key. At the very least the top and bottom of the connector should be clearly marked as such.



Testing

When testing the unit it is helpful to connect a potentiometer of about 5 or 10 kilohms in value to analogue input 0, as shown in **Fig 5.** If the program given below is run it should be possible to adjust the returned value from 0 (fully anticlockwise) to 255 (fully clockwise), with intermediate settings giving intermediate values.

```
10 CLS
20 ?&FC01 = 0
30 ?&FC00 = 0
40 PRINT TAB(19,12); ?&FC00
50 GOTO 30
```

Line 20 selects analogue input 0, and this input will be connected through to the converter until a new value is written to ?&FC01. If you write (for example) 3 to this address the returned value should be 0, and the potentiometer should have no effect unless its wiper terminal is transferred to analogue input 3. Line 30 provides the "start conversion pulse", and line 40 PRINTs the returned value at roughly the centre of the screen. Line 50 simply loops the program back to line 30 so that a continuous stream of readings are taken. Note that the "start conversion" pulse must always be sent before the converter is read.

The following program can be used to test digital input F1:

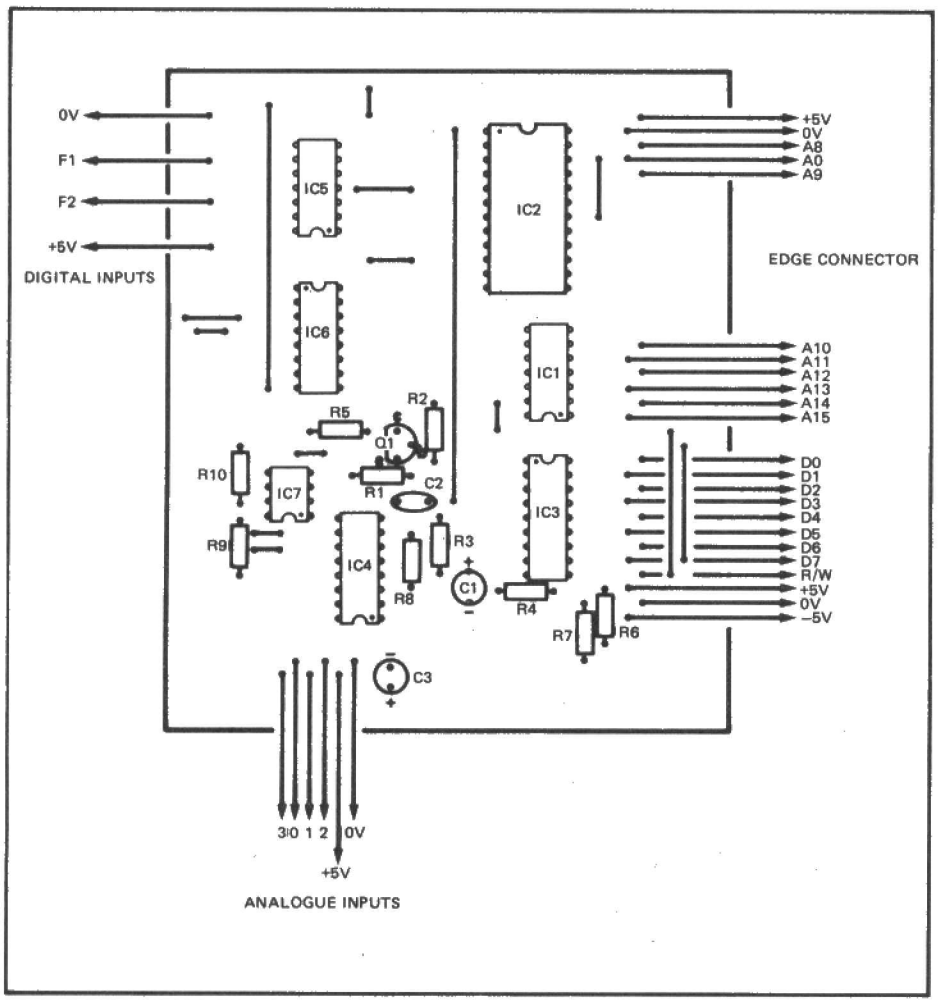
```
10 CLS
20 PRINT TAB(19,12); ?&FC01 AND 1
30 GOTO 20
```

This will normally return a value of 1 as the input will float to the high state, but connecting F1 to the 0 volt supply should give a reading of 0. Input F2 can be read by using "AND 2" at the end of line 20 instead of "AND 1". The value normally returned will then be 2, with a reading of 0 if input F2 is connected to the 0 volt supply rail. Some "fire-button" inputs are edge sensitive, but in this case the value returned is a true reflection of the logic state at the input.

As mentioned earlier, the status output of the converter is irrelevant when BASIC is being used, and it cannot meaningfully be read from BASIC (it will always be at logic 1).

◀ **Figure 5. Connecting a potentiometer to the port, for testing purposes.**

▼ **Figure 4. Details of the printed circuit board.**



PARTS LIST

Resistors (all 0.25W 5%)

R1, 4	390R (2 off)
R2, 5, 9	1k (3 off)
R3	82k
R6	1M8
R7	8k2
R8	6k8
R10	10k

Capacitors

C1	1uF 63V radial elect
C2	100pF ceramic plate 2%
C3	100uF 10V radial elect

Semiconductors

IC1	74LS30
IC2	74LS154
IC3	ZN427E
IC4	4051BE
IC5	74LS125
IC6	74LS174
IC7	NE555
TR1	BC239 or BC109

Miscellaneous

Printed circuit board, Ribbon cable, 2 x 25 way 0.1 inch edge connector, DIL IC sockets (1 x 24 pin, 1 x 18 pin, 2 x 16 pin, 2 x 14 pin, and 1 x 8 pin), Veropins.

THE BBC MOS

More important than BASIC?

Mike James explores the finer points of the BBC MOS

The BBC Micro and now the new Electron most often receive praise, and the occasional criticism, on behalf of the version of BASIC that they run. BBC, or Acorn BASIC, is certainly the most visible software component of the two machines but is not the only one and it may not even be the most important. Of the 32K of ROM used in both machines 16K is allocated to the built in BASIC and 16K is allocated to the Machine Operating System or "MOS" as it is usually called. The fact that both the BASIC and the MOS take the same amount of ROM might suggest that they are of roughly comparable complexity. The fact that the BASIC ROM was finalised long before the MOS ROM should suggest that the MOS is if anything more complex than the BASIC!

The reason why there is so little comment on the MOS probably has something to do with it being an operating system. The purpose of an implementation of a language such as BASIC is obvious but the purpose of an operating system is not so clear. The quality of a BASIC is therefore easy to judge but it is difficult to set a standard for an operating system. Tradition has it that the purpose of an operating system is to make the facilities of a machine available to other programs and ultimately the user and this is indeed what the MOS does. However, if an operating system offers up a facility in such a way that it is difficult to use does the fault lie in the operating system or the hardware of the machine? You might think that the answer to this question depends on the particular case, sometimes the software would be at fault and sometimes the hardware. The surprising thing is that it is *always* the software! No matter how horrible the hardware is it is possible for the software contained within the operating system to deliver it to the user in a way that is easy to use. The only thing that you can blame the hardware for is lack of speed and this might make the software's job pointless – an easy to use but incredibly slow system is not something that anyone wants to use! An ideal operating system should, so to speak, "tame the machine's hardware" so that other software can concentrate on putting it to good use.

Another reason why operating systems are not an important factor in personal computing is that machine designers have often overlooked the need for good operating systems and concentrated on the quality of the BASIC. This has forced the BASIC grow extensions that look after whatever extra hardware that the machine has. An example of this approach at its most extreme can be seen in the APPLE II's so called Disk Operating System or "DOS". This is so much an afterthought to APPLESOFT BASIC that it is difficult to see it as a separate piece of software! The BBC Micro was possibly the first machine to fully use an operating system both to make facilities available and to enhance the performance of the hardware. It is difficult to put forward a case to say that Acorn succeeded in producing a good overall design for the MOS without going into a great deal of detail concerning the BBC Micro's hardware and the different ways that it could be used. However, it is not difficult to pick on a number of examples that show how the MOS improves some aspect of the hardware or makes it easier to use the hardware. But first it is worth looking at the general principles that lie behind the MOS.

An Advanced Design

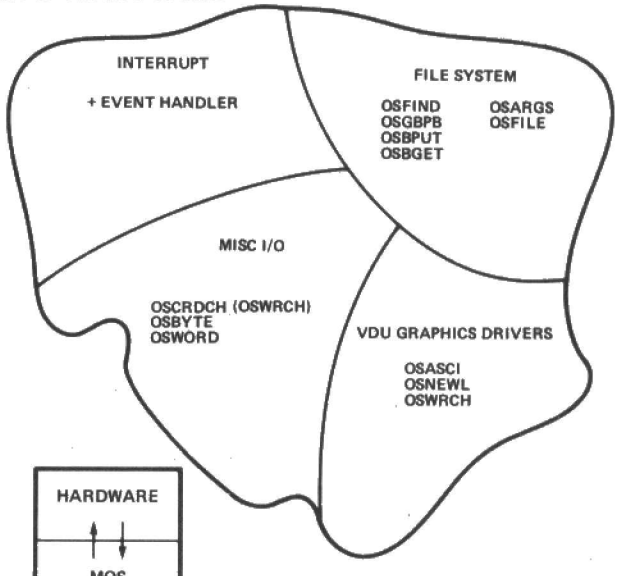
The main design decision that seems to have given rise to the MOS is that all of the machine's I/O would be handled by it, and it alone. In this sense the MOS can be thought of as a layer of software that separates other system software such as BASIC from the hardware, (see **Fig 1**). This separation enables languages such as BASIC to be hardware independent. At first it may be difficult to see any payoffs for the user of this hardware independence but it is, for example, responsible for the uniform way that files are handled no matter what the file device concerned – tape, disk, network or telesoftware. As far as file devices are concerned there is just one set of I/O commands and the variability in the storage device is 'absorbed' by the filing system software.

Miscellaneous and character oriented devices, such as the sound generator and the serial interface, are always difficult to build into an operating system in a regular way but the BBC MOS does its best. The text and graphics display is such an important part of the machine that it is given a section of the MOS all to itself – the VDU drivers. Other I/O devices are combined with miscellaneous operations and dealt with in two classes – those that only require a small amount of information to be passed and those that require a large or variable amount of information to be passed. The reason for this division is entirely practical because it enables a simple method of communicating with the MOS to be used wherever possible.

The final, and perhaps the least obvious, action of the MOS is to look after interrupts. The BBC Micro and the Electron are both advanced machines in that interrupts are an essential part of the way they work – not, as in the case of so many machines, an afterthought added to control a special I/O device or provide a clock. An interrupt-driven machine has a very special 'feel' about it for both programmers and users. Apart from providing the real time clock in the pseudo variable TIME the interrupts are used to service and maintain the extensive system of I/O buffers and queues.

The structure of the MOS as described above can be seen in **Fig 2,** along with the names of the machine code subroutines that BASIC and other software uses to communicate with each section. Notice that although the interrupt and event handler looks as though it has no way of letting either BASIC or a user program gain access to it, this is not the case, as will be explained later. As with any piece of real software there are always odds and ends that don't fit into the overall classification and the MOS is no exception. For example, there is a 'command line interpreter' which, while not essential, does make the MOS easier to use. Now that the structure of the MOS is clear it is time to look more closely at each of its parts.

Figure 1. The flow of data.



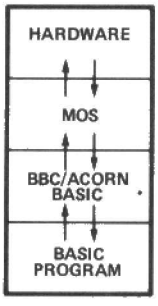


Figure 2. The structure of the MOS.

The File System

In most computer systems it is useful to distinguish two types of I/O devices – those that work in terms of named files and those that work by transferring small units of data one after the other. File devices include the well known cassette recorder and floppy disks of all kinds. However, for the BBC Micro the range of file devices extends to

exotic systems such as networks, telesoftware and ROM cartridges. Such diverse systems might lead you to believe that a different method of handling is appropriate and indeed necessary for each one. This is not so! The idea of a file and the operations that manipulate it are independent of the actual physical device used to store it. The physical device affects the speed of storage and retrieval (and very occasionally it may not allow a particular operation) but it doesn't alter the basic principle of handling files.

The MOS uses a standard set of six machine code subroutines for all file operations no matter what type of device is in use. Two of the subroutines, OSFIND and OSARGS, are concerned with general file operations such as 'opening', 'closing', 'deleting' etc. The remaining four are used to read and write files. OSBGET and OSBPUT are used to read and write a single byte from and to a file that is already open. OSFILE reads or writes an entire file in one operation. The final subroutine OSGBPB will write or read a variable sized block of bytes to the file – this can only be used with devices that can sensibly support random access such as the disk system.

Each one of these subroutines is used by referring to a fixed location in the MOS ROM but, as these locations immediately transfer control to jump vectors in RAM, the position of the code that actually carries out the operation can easily be changed. In this way the code used when you call one of the subroutines can be made appropriate for the currently selected filing system – tape, disk, network etc. These subroutines are used by BASIC to implement its file I/O and you should be able to recognise the correspondence between the BASIC commands GET, PUT and LOAD/SAVE with the MOS subroutines OSBGET, OSBPUT and OSFILE. Of course the more familiar INPUT and PRINT make repeated use of the MOS subroutines to handle their multiple data transfers.

The VDU Driver

The VDU driver is responsible for performing all of the text and graphics operations allowed on the BBC Micro or Electron. If you know how the BBC Micro and Electron produce their TV display you will realise that the VDU driver has a lot to do simply to make a character appear on the screen. As well as this fundamental job it also has to provide the colour control, implement the graphics commands, look after the scrolling, implement and maintain the text and graphics window and look after just about every other feature of the display. With so many different things to do you might expect that the VDU driver would consist of a great many different subroutines each with a different name and method of use. As in the case of the file system it is possible to make things simpler by careful planning and observing how information is normally generated and sent to a video display. As text is usually sent to the display in the form of a stream of ASCII codes it makes sense to continue to use the same method to communicate all the data to the VDU driver. In other words, the VDU driver can use a single entry point as long as it examines the stream of ASCII codes that is being sent to it for special 'control codes'. This is indeed what happens. The single entry point is OSWRCH and the ASCII codes are extended to include all of the text and graphics operations. For example, if the VDU driver detects an ASCII code of 12 then it will clear the text screen, a code of 16 will clear the graphics screen and so on. Sometimes the amount of information needed for a text or graphics operation is greater than a single ASCII code can convey. In this case a sequence of ASCII codes is required. For example, following ASCII code 31 the next two codes are not interpreted in the usual way but are taken to be the new X and Y position of the text cursor.

The familiar BBC/Acorn BASIC text and graphics commands are implemented by sending the appropriate ASCII codes to the VDU driver. For example, the command CLS results in the code 12 being sent to the VDU driver. In this sense the BASIC statements:

```
      CLS
      PRINT CHR$(12);
and   VDU 12
```

all achieve exactly the same results, they send code 12 to the VDU driver and so clear the screen and are therefore the same! TAB(X,Y) first sends code 31 and then two codes whose values are given by X and Y respectively.

In practice it is useful to have three entry points to the VDU driver for convenience – OSWRCH which sends the code stored in the A register to the VDU driver, OSASCI which inspects the code and if it is a carriage return sends a newline code to the VDU driver, and finally OSNEWL which is an entry point that automatically sends a newline followed by a carriage return code.

The power of this simple system of using a stream of ASCII codes to control the complex text and graphics operations takes some time to appreciate. One of its advantages is that both the BASIC programmer and the assembly language programmer control the display in exactly the same way. The sequence of codes that the BASIC programmer sends to the VDU driver, using say the VDU command, is exactly the same as the set of codes that the assembly language programmer has to send to the VDU driver by directly calling OSWRCH.

Miscellaneous I/O

There are two machine code subroutines that control a wide range of MOS activities– OSBYTE and OSWORD. Exactly what OSBYTE and OSWORD do is controlled by the code stored in the A register before they are called. Once again the programmer is spared from having to deal with a large number of separate and specialised subroutines by the simple expedient of using a single entry point and an action code. OSBYTE is made available to BASIC programmers in two ways. Firstly, it is used to implement many BASIC commands. For example, the INKEY function calls OSBYTE to read the keyboard. Secondly, many OSBYTE actions are available via the familiar *FX command. Only OSBYTE actions that do not return information to the program can be used via the *FX command, because, although the *FX command can transfer data to OSBYTE, it has no mechanism for returning it. The command *FX a,x,y places the value 'a' in the A register, 'x' in the X register and 'y' in the Y register and then calls OSBYTE. In this sense it is exactly equivalent to the assembly language –

```
LDA a
LDX x
LDY y
JSR OSBYTE
```

If x or y are omitted the command will automatically load the appropriate registers with zero.

The OSWORD subroutine works in a very similar way to OSBYTE but as, the amount of data involved is more than just the A, X and Y registers can hold, the *FX or an equivalent cannot be used to access OSWORD from BASIC. The only access to OSWORD that a BASIC programmer has is indirect and via the BASIC commands that call OSWORD to implement their operations. For example, the SOUND command sets up a data block of information about pitch, volume etc. It then loads the A register with 7, sets the X and Y register to point at the data block and jumps to OSWORD which does the real work of setting up the sound generator.

The *FX commands are well documented in the BBC Micro and Electron's manuals and so it isn't worth repeating the list here. However, there are a number of OSBYTE and OSWORD calls that are useful from BASIC but no direct command is provided to access them. For example, the OSWORD call with A set to 10 will return the dot definition of any character. The ASCII code of the character is stored in the first location of a data block and the dot pattern is stored in the following eight bytes of the block when OSWORD returns. Although this OSWORD call isn't available in BASIC it is easy to write a procedure to implement it –

```
1000 DEF PROCshape(CODE%)
1010 ?&600=CODE%
1020 A%=10
1030 Y%=&06
1040 X%=&00
1050 A%=USR(&FFF1)
1060 FOR A%=1 TO 8
1070 DOT%(A%)=A%?&600
1080 NEXT A%
1090 ENDPROC
```

The eight bytes of the dot pattern for the character corresponding to CHR$(CODE%) are returned in the array DOT% which must be DIMensioned in the main program before PROCshape is used. The area of memory starting at &600 is used for the data block because it is unlikely to be in use for anything else. It is normally used for the parameter block for a CALL statement. As another example the OSWORD call with a set to 11 returns the logical to physical colour assignment as set by any previous VDU 19 commands. This is also ▶

not available as a direct BASIC command but it is easy to add it as a function –

```
2000 DEF FNphyscol(L%)
2010 ?&600=L%
2020 A%=11
2030 Y%=&06
2040 X%=&00
2050 A%=USR(&FFF1)
2060 =?&601
```

This will return the physical colour code assigned to the logical colour code stored in L%. You should be able to see the similarity between this function and the previous procedure.

The final two I/O procedures – OSRDCH and OSWRCH (again) – are used to read and write single bytes of data to any of the character oriented devices. Normally OSRDCH will return the ASCII code of any key pressed on the keyboard and OSWRCH will send the ASCII code of a character to the VDU driver. This is because the keyboard and the screen are the default I/O devices. It is possible to change this default selection using the OSBYTE subroutine and in this sense OSRDCH and OSWRCH are general purpose single character input and output routines. For example, following *FX 3,3 OSWRCH sends a character to the printer port and following *FX 2,1 OSRDCH gets characters from the serial port.

Interrupt and Event Handler

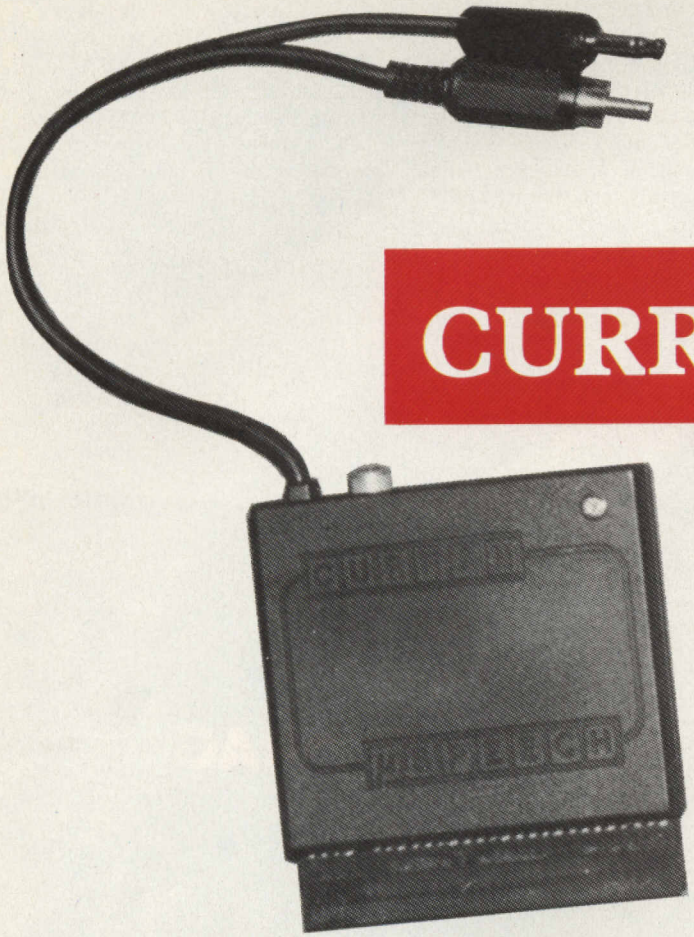
While the BBC Micro is running interrupts are generated by the interval timer responsible for maintaining the pseudo variable TIME. Each time one of these regular interrupts is received the interrupt handler, whose address is stored at &204, is called. This not only updates TIME it also checks to see if anything else needs doing. For example, it is responsible for altering the parameters of the sound generator for the duration of a sound countrolled by an envelope. It is this action that makes the BBC Micro's sound generator so powerful. From a hardware point of view it is not at all impressive but, when it is added to the frequent and periodic attention that the interrupt handler provides, it is as good as more complex hardware and far more flexible.

The interrupt handler also looks after the system of queues that are so important a feature of I/O through the MOS. Using the sound generator as an example for the second time it is easy to see that at each interrupt the interrupt handler has to check to see if the current sound has reached its specified duration or not. If it hasn't then it the sound generator is left alone. If it has then the sound queue associated with that channel is inspected. If it's empty then the channel is switched off otherwise the parameters stored in the queue are used to start the next tone and the queue is 'moved up by one'. This interrupt servicing of the sound queue is responsible for the BBC Micro apparently being able to do two things at once – run a BASIC program and generate a sound. The printer and serial ports are both associated with queues or buffers that are serviced by the interrupt handler and, as in the case of the sound generator, this greatly enhances the performance of the entire system. For example, it is quite possible for a BASIC program to print out a large quantity of data while getting on with another job. All it has to do is send data to the printer buffer until it detects, using ADVAL(−4), that it is full. Then it can get on with another job and the interrupt handler will send a character at a time to the printer, at the rate that the printer can accept. Of course, if left in this state, all that would have been printed is a single buffer full of data but if, while carrying out its other jobs, the BASIC program occasionally checks to see if there is any space in the buffer, again using ADVAL(−4) and transfers enough data to keep it topped up any amount of data can be printed without stopping the BASIC program unnecessarily.

Overall Design

There are many good points that could be displayed as evidence that the MOS is an excellent piece of software but in many ways it is the way that the whole system fits together that is impressive. It is a relief to discover system software on a microcomputer that gives a hint that it was designed rather than thrown together and then finalised by adding sections to solve specific problems. The MOS is certainly a well designed and implemented operating system suitable for a micro computer and it is clear that a lot of thought went into its construction.

E&CM



CURRAH μSPEECH

Is the μspeech with its ULA based text-to-speech interpreter a breakthrough in low cost speech synthesis? Gary Herman has the answer.

The Microspeech is specifically designed for the Sinclair Spectrum. The unit is neat and packaged in a small plastic case some 8cm x 7cm x 1.6cm. The case is glued together and cannot, unfortunately, be opened for inspection. It plugs into the expansion socket at the back of the computer or into Sinclair's interface unit – although it cannot be 'chained' with other devices since there are no bus lines coming out of it. The unit takes power from the Spectrum's expansion port and the computer **must** be turned off before plugging it in. The unit features two leads, one of which plugs into the TV socket on the back of the Spectrum and the other into the MIC socket. With the aerial lead from your television plugged into the Microspeech itself, sound generated by the unit and by the computer will be mixed and fed to the television. It would have been better if these leads had been slightly longer – at around 24cm there is very little 'give' when the unit is plugged in. In order to record speech or play it through an external amplifier, a lead connected to the Spectrum's MIC socket (the line lead) may be used.

Unlike most other speech synthesis units in the same price range, the Microspeech contains a text-to-speech interpreter. It uses an 'allophone' system based on the General Instruments SPO256-AL2 speech chip – which is similar to, but not compatible with, the commoner SC-01 chip from Votrax.

About Allophones

Allophone speech generation uses units of spoken sound, rather like phonetic syllables, in combination to produce words and phrases. Thus the word 'house', for example, can be formed from the allophones H, UH3, AH2, U1 and S. As you can see (and as a little thought will demonstrate) the alphabetic and phonetic spelling of a word are rarely the same. Incidentally, some of you may be more familiar with the term 'phoneme' than you are with 'allophone' – however, they are not strictly the same thing if only because the set of phonemes is considerably larger than the set of allophones.

Most cheap allophone speech synthesis systems make use of little more than the bare speech chip. Commonly, such a chip contains 64 allophones – including pauses of various lengths – which are accessed by a 6-bit code consisting of the six least significant bits of a byte. Thus, the allophone SH is coded 00100101 (or 37 decimal) on the Microspeech's chip and 00010001 (or 17 decimal) on the SC-01. To produce the sound, a chip has to be connected up to a filter and the appropriate code sent to it. The simplest allophone systems are treated as output devices and a BASIC program to generate speech then takes the form of instructions to output numerical data via the computer's I/O port.

Text-to-speech interpreters are more-or-less complicated algorithms which remove the necessity of constantly looking up figures in a table of allophone codes and no less constantly revising your opinion of how a given word should be spelt phonetically. In their most sophisticated implementations, they translate words into sequences of allophones using data on the common pronunciation of individual letters, groups of letters and even whole words.

The Microspeech's text-to-speech interpreter is not that sophisticated. It will recognise and pronounce individual letters but goes no further. Thus it will mispronounce 'through', 'good', 'comely' and even 'hello' although it should do alright with 'red', 'bad' and 'electronics'. You will have to spell 'love': L-U-V (shades of the Shangri-Las)!

Speaking Out

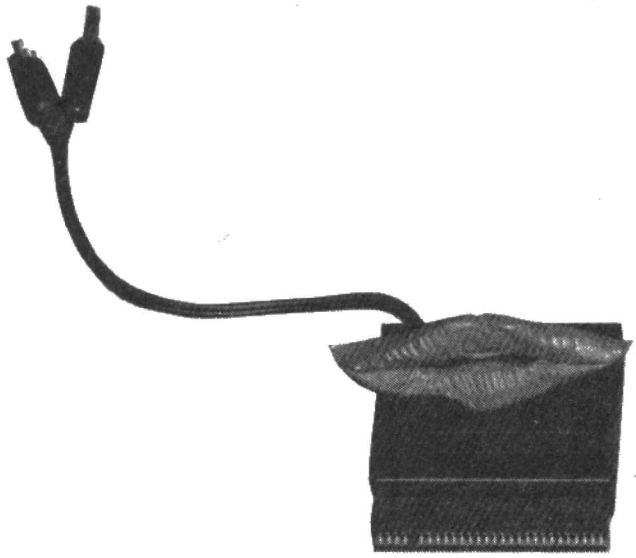
For the impatient among you, the Microspeech is ready to talk as soon as it's turned on. Press the ENTER key on your Spectrum and the machine will respond with the word 'enter'. Keep the key pressed and the word will repeat at a speed determined by the key repetition rate POKEd into the appropriate location in the Spectrum's memory. Since most keywords and printable characters are voiced it's possible to have some fun changing the repetition rate and making the Microspeech say 'G . . . G . . . G . . . GOTO' or whatever. Fortunately, typing 'LET keys=0' will turn of this facility which can after a while become extremely annoying (but is obviously valuable to people with impaired vision who would like to use a computer). 'LET keys=1' will turn the key-voice back on.

The simplest method of entering speech in a program is to use the Microspeech's text-to-speech interpreter. Words and phrases are entered as strings with the reserved variable name: s$. Phonetic spelling should be used, so that:

```
LET s$="he(ll)(00)"
```

will sound 'hello'. But phonetic spellings are not fixed – LET s$="hullo" gives an acceptable 'hello' and, of course, imaginative phonetic spellings can be used to produce a range of accents.

The contents of the variable, s$, are read and entered into a 'first-in first-out' (FIFO) buffer initialised as 256 bytes on top of the Spectrum's top of BASIC RAM address (which is, therefore, moved down 256 bytes). The size of this buffer can be increased or decreased by use of a CLEAR command which makes RAMTOP point to a specified address, with the proviso that the buffer must be at least 6 bytes to contain data on buffer and system status. An attempt to invade these top six bytes will result in the system crashing and re-initialising.



Data is stored in the buffer until the system requests its transfer to the Microspeech sound chip and each time a LET s$=' command is read the data is entered into the buffer. To prevent the data being corrupted and to give the buffer time to clear when it gets close to full, each 'LET s$=' command must be followed by a PAUSE command. Words and phrases can be entered into other strings beside s$ and then transferred to s$. In this way, words and phrases can be concatenated and sliced according to Spectrum BASIC's string-handling routines.

PRINTing s$ each time it occurs not only lets you see what is being spoken but also reports on the status of the speech-string. If the string is entered into the buffer, a 'PRINT s$' will return the string with an asterisk in place of the first character – '*e(ll)(00)'. If the string is unacceptable (because, say, it includes some uninterpretable symbols) 'PRINT s$' will return it with a question-mark in place of the first character – '?e(ll)2oo)', for example. If the buffer is full, the string will be ignored and 'PRINT s$' will return it uncorrupted.

The buffer can, of course, be addressed directly from a BASIC program by using a POKE command. In doing this, you must also update the buffer pointer which is contained in the top six bytes of the buffer (at 65365 and 65364 on a 48K machine). The Microspeech manual – a brief but happily informative document – gives an assembly language routine for doing this.

Swan Song

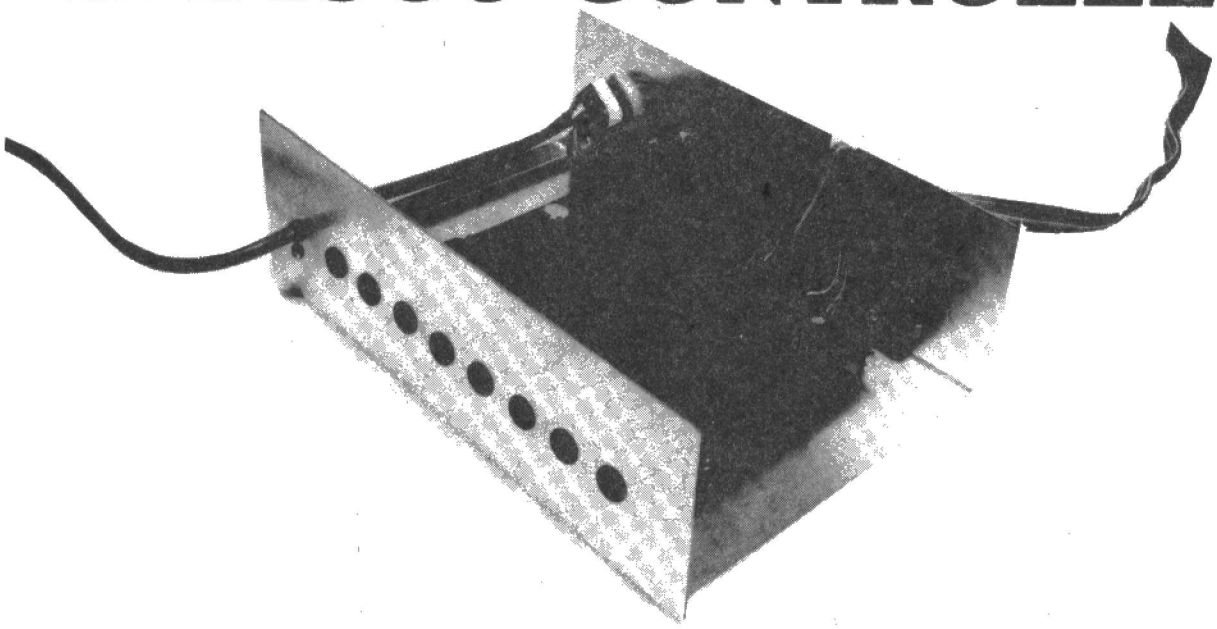
Finally, the Microspeech has one quirk that may be of interest. Because the Spectrum's CPU is not interrupt managed, when certain commands are entered the Microspeech's buffer cannot be updated (nor, indeed, can any other task be undertaken). This is most obvious and irksome when it comes to the BEEP command which halts all program execution until the BEEPed sound is over. BEEPing in the middle of an allophone draws out the allophone so that it and the BEEPed tone sound together for as long as the tone lasts. The effect is not unlike singing . . . although, to be truthful, it's not much like it either.

The Microspeech is an impressive piece of equipment, given the limitations of price and the Spectrum. Like all low-cost speech synthesisers, it has the unhappy knack of sounding like a retarded Dalek with a cold, but I found it well-documented, relatively easy to use and reliable. These are not qualities necessarily apparent in considerably more expensive systems.

The text-to-speech interpreter is, all things considered, a minor miracle. It would have been nice for some more technical information on the circuit and some more examples of phonetic spelling to have been provided. Although there is a facility for adding stress (by using capital letters) and creating 'drawls' (by repeating certain allophones), there is no immediate way of filtering the device's output. It may be possible to filter the output from the line lead (could be worth experimenting?), but perhaps Currah will put their minds to it. Until then, we'll just have to settle for robotic male voices. (Currah Computer Components Ltd., Graythorp Industrial Estate, Hartlepool, TS25 2DF).

E&CM

ZX DISCO CONTROLLER



Last month Mark Stewart described the hardware of our ZX Disco Controller. This month he moves on to discuss the software.

In its basic form the unit can be treated like a simple 8 bit output port. Setting a bit to 1 switches on the lamps. Setting a bit to 0 switches off the lamps in the associated bank. The testing routine described previously demonstrates a simple on/off procedure. Note that once set the state of the lamps remains unchanged until the computer issues another instruction. In effect the 8 bit latch chip IC3 acts as a one byte static RAM.

More Sophistication

The short programs shown below are for the ZX81. For the Spectrum the POKE instructions will need to be replaced by OUT instructions. Also the timing loops will need to be adjusted because the Spectrum and ZX81 run at different speeds. All of the programs are based on simple on/off instructions. The mains zero cross switching should be on to minimise interference.

The next program uses the computer to control the timing of the triac turn on pulse in each half cycle of the mains. It demonstrates the technique of "phase control" which can be extended into much more ambitious programs which control the brightness as well as the sequence of the lamps. **Fig 6** shows the function of the two principal timing loops in the program in relation to the AC mains waveform. For programs of this type full control of the triacs is essential and SW1 must be set for random triggering. The program is written entirely in machine code for the ZX81 and must be run in fast mode.

The program occupies 79 bytes. It is stored in a REM statement. The data loading program should first be entered. This is the usual loader program, except that decimal codes are being used instead of the more usual Hex. Line 20 asks for each op code to be entered.

The op codes are entered one by one from the OP-CODE (decimal) column of the main program. Read from left to right line by line.

When the program is running all eight channels will be controlled by the top row of keys on the ZX81. Use the left hand keys for dimmer, and the right hand keys for brighter. To return to BASIC press any key on the left side of the bottom row except shift.

The program comments with the main listing give some idea of the operation. Two timing loops are involved. The main loop sets the timing of the triac trigger pulses in the mains cycle. A second loop sets the width of the trigger pulse – the Gate pulse timing loop. **Fig 6** shows the operation graphically. To brighten the lamp the main timing loop is shortened so that the triacs are triggered earlier in the mains cycle. Dimming the lamps is the opposite.

Using this principle the control can be extended to individual lamp control, automatic fading and so on. It is all a matter of software.

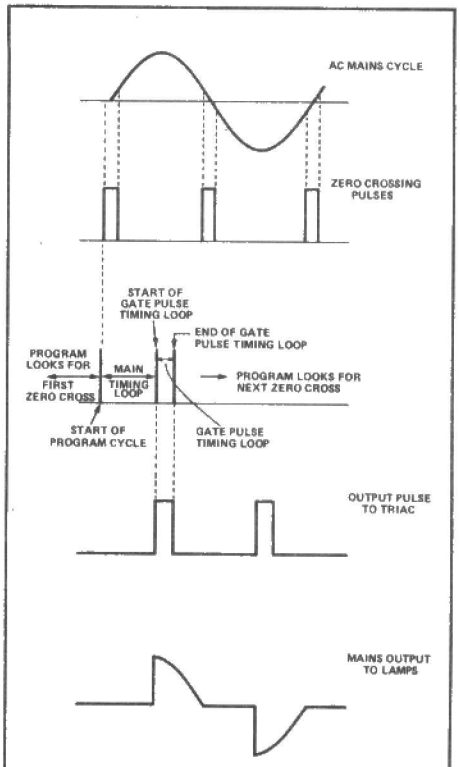


Figure 6. The two timing loops of the dimming program are related to the mains AC waveform.

ZX PROGRAMS

RANDOM PATTERN GENERATOR:-

```
10 POKE 8192, RND*255
20 PAUSE 10
30 RUN
```

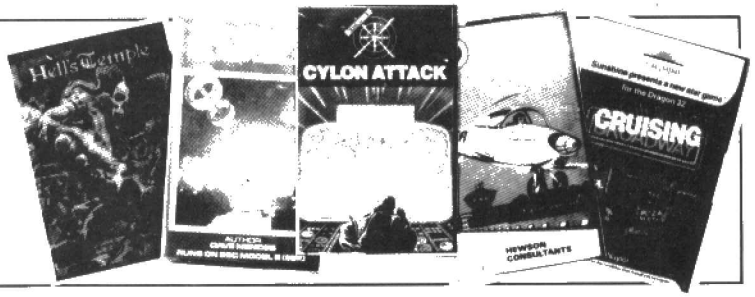
SINGLE BULB CHASER:-

```
10 LET A = 1
20 POKE 8192,A
30 PAUSE 3
40 LET A = A*2
50 IF A ÷128 THEN RUN
60 GOTO 20
```

STROBE EFFECT:-

```
10 POKE 8192, 255
20 PAUSE 5
30 POKE 8192,0
40 PAUSE 25
50 RUN
```

SOFTWARE SELECTION



As a concession to the fact that the Christmas season is fast approaching, and that our readers may wish to swap more weighty pursuits for some lightweight enjoyment during this time, we look at a range of current games software. This month we look at packages for the Spectrum, Dragon, Oric and BBC/Electron.

DRAGON 32

And All Because

The idea for this game comes from the long running commercial campaign that features an athletic gentleman going through various rigorous manoeuvres in order to get a box of chocolates to a mysterious lady.

This graphic game consists of nine different screens each requiring the player to negotiate various obstacles (including killer mushrooms) while 'driving' a variety of vehicles that include everything from a horse through motorbike to a hang glider.

The game features a high score table and is certainly original in concept.

B&H Software, 208 King Street, Cottingham, Hull.
Price £6.95.

Games Pack 1

A collection of games that comprise 10-Pin Bowling, Muncher, Micropoly, Mastermind, Noughts and Crosses, Wells of Omicron, Simon and Kings of the Valley.

The games are written in BASIC and all the restrictions in speed and quality of graphics that this implies are quite apparent. Whether or not the variety of games offered makes up for the lack of sophistication of the individual titles is difficult to assess but on balance the package seems to offer value for money particularly as the programs can be listed to allow the techniques used to create the games to be analysed.

Abacus Software, 21 Union Street, Ramsbottom, Nr. Bury, Lancs.

Frogger

A faithful emulation of the Arcade game that involves steering a frog across a five lane highway and then a five lane river to the safety of one of five froggy homes. En route there are plenty of nasties that have squishing said frog as their sole aim in life.

This package is produced under licence from the American originators of the arcade game and thus has the advantage of being very close to the arcade offering.

As is usual in Microdeal packages such sophistication as joystick or keyboard option, fast and slow speeds, one or two player option, high score and freeze frame are all offered and add to the game's attraction.

If you're a Frogger addict, this version won't disappoint.

Microdeal, 41 Truro Road, St. Austell, Cornwall.
Price £8.00.

Hide And Seek

This game is based on the traditional 'Kim's Game' which, for those of you not too sure of your children's games aims to develop short term memory – or that's the psychologist's story at any rate.

The game starts when various objects are put into boxes, which then 'close'. The object of the game is to decide/remember just exactly what is in the various boxes.

Another game on this tape is 'What's Missing'. The basic idea is the same but this time, after hiding the objects, the boxes open to reveal the fact that one object is missing. The task this time is to identify the missing item.

This game, while being reasonably entertaining to play, is very much aimed at educating children through play.

Dragon Data, available from local stockists.

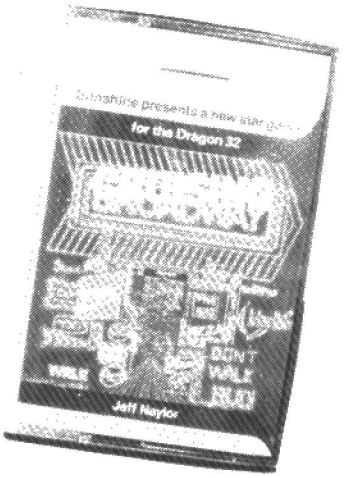
Nightflight

Nightflight demands that the player masters the skills involved in flying a single engine light aircraft. The object of the game is to take off, fly the plane around for a while and then to land preferably, as the instructions put it, on the runway.

As may be expected from Salamander, the software, the game is of a high standard with many of the plane's parameters under 'pilot' control and a comprehensive array of the aircraft's systems, including altitude, fuel and speed, displayed on screen.

The program goes as far as allowing the pilot to go into Barrel Rolls and to perform loops – this is not recommended for the tyro pilot however.

Salamander Software, 27 Ditchling Rise, Brighton, East Sussex, BN1 4QL.



Cruising On Broadway

This is a fairly straightforward game that makes little demand on the intellect but sets out to test the manual dexterity of the player. The game involves piloting a character along a maze of straight lines while avoiding the attention of a semi-intelligent Chaser that will destroy your character if you do not keep a respectable distance between the two characters.

The game progresses through a series of mazes of increasing complexity and, from the third level, there will be two Chasers after you and things will be that bit trickier.

Sunshine, 12-13 Little Newport Street, London WC2

Dragon Fly

Another Dragon flight simulation package that features a full instrument display which includes Heading, Bearing, Artificial Horizon, RPM and Flaps. Two runways are featured and the package allows the player to indulge in a certain amount of aeronautics.

Hewson Consultants, 60A St. Mary's Street, Wallingford, Oxfordshire OX10 0EL.

Sinclair Spectrum

Strike Attack 48K

The object of this game is to pilot your jet (a Phantom) through a barrage of enemy fire in order to reach your target. The game does not aim to provide a flight simulator as such and the on-screen information is quite sparse being limited to that of altitude, speed and a warning of any enemy action in your airspace.

The game is not easy to master and games players looking for a challenge might find it interesting.

Micro-Mart, Greenhill Ind. Estate, Kidderminster, Worcester
Price £7.95

The Quill 48K

A number of packages that make it possible to write in machine code without in fact having any knowledge of machine code have appeared in recent months. The Quill is such a product and it is designed to provide the user with an environment in which machine code adventure programs can be readily written.

The package is accompanied by a voluminous 52 page manual that for the first half takes the user through an explanation of just what an adventure game is and how the Quill can be used to create an original game.

The second half of the manual provides a detailed description of the Interpreter, Database and Editor as well as an explanation of the Error Messages.

Adventure games are absorbing in themselves and anyone keen on these sort of games should find 'Rolling Their Own' a rewarding pastime.

Gillsoft, 30 Hawthorn Road, Barry, South Glamorgan, CF6 8LE.

Doom Bugs

This is not a game for anyone who likes to keep spiders at arms length. The game is based around a great big bug and its pot of honey. The cast also includes a host of other bugs intent on eating either our hero bug, the honey or both.

The aim is to get the hero out of the trap in which we find it by chomping away at a trap door to get to the next level.

The game is written by Jim Scarlett who wrote Jaws Revenge and is likely to appeal to ardent gamers one and all.

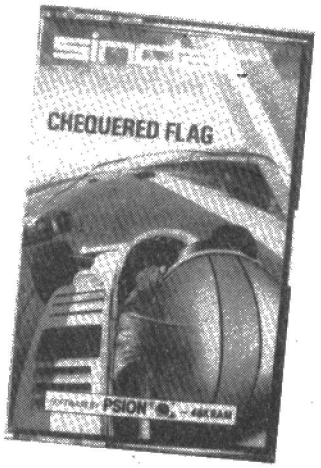
Work Force, 140
Work Force, 140 Wilsden Avenue, Luton, Beds.
Price £5.50.

Games Designer 48K

This package allows one of eight pre-programmed games to be customised according to the users want. Virtually all of a games standard ingredients can be altered from the Game Format – the direction of movement and screen area occupied by the game elements, through the colours of both foreground and background, the nature of the sound effects to the way in which the attack waves are configured.

This package follows the current vogue of offering inexperienced programmers the chance to design their own games without having to have a detailed knowledge of machine code techniques. The interactive nature of this package adds to its appeal and means that anybody can tailor one of eight games to their exact requirements.

Software Studios – distributed by Quicksilva Ltd., Palmerston Park House, 13 Palmerston Road, Southampton, Hants SO1 1LL.



Chequered Flag 48K

This is one of the latest Psion/Sinclair products and is described as a companion to the best-selling Flight Simulation package from the same stable.

The game starts after the player has made a selection between three different cars with varying power and handling characteristics. An instrument panel, a la Flight Simulator, is provided and the requirement to monitor this while keeping at least one eye on the road provides a real challenge.

We have it on good authority that the way in which a car that attempts to take a corner at too fast a speed is based upon real grip and centrifugal force measurements.

Sinclair Research, major retailers
Price £6.95

Space Raiders

This is one of the new ROM based software packages for use with Interface 2.

This game is by today's standards slightly old fashioned being a version of the very first arcade games to appear in this country. It features the familiar rows of alien invaders that slowly work their way down the screen. You are in charge of a laser base which can be moved only in the horizontal plane.

The only shelter provided is a few buildings in which you can temporarily hide your laser base.

If you have fond memories of those early space invader machines – this is a game that you'll enjoy.

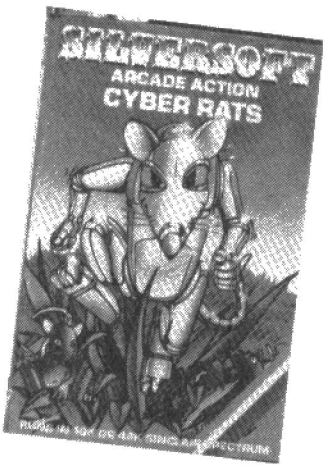
Sinclair Research, major outlets.
Price £14.95

Heathrow Air Traffic Control 16K/48K

While there are plenty of packages about that will appeal to any budding pilots this product is aimed more at those whose aspirations are in the area of air traffic control.

The program features extensive information on the topography of the land around Heathrow and, once the 11 pages of instructions provided with the game have been mastered, anyone with an interest in aviation can expect to spend many absorbing hours with the simulators.

Hewson Consultants, 60A St. Mary's Street, Wallingford, Oxfordshire.



Cyber Rats

An arcade game that involves blasting metallic menaces into the next galaxy. Player controls allow movement in both the vertical and horizontal directions and the game features a hi-score display and a two player option.

Silversoft, 20 Orange Street, London WC2H 7ED.

BBC Micro

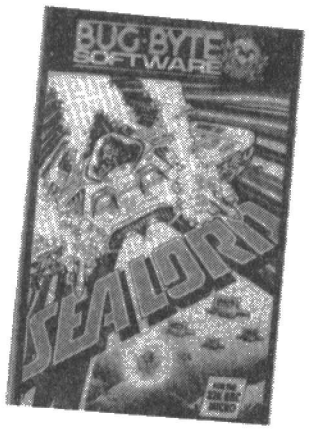
Missile Control from Gemini is definitely one of the best new games for the BBC micro. It is a devil to load the second part of the program, but after the third or fourth attempt you realise it was worth the effort. The object is to save a city from annihilation by incoming missiles, from a second wave of more elusive planes and satellites, and from wave 6 (I never got that far) of 'Smart' missiles which can cleverly avoid the ground fire. The players anti-missile missiles are fired from a choice of three batteries and aimed by a cross on the screen which can be moved by the cursor keys. This game really requires joystick control however, because there are far too many (badly chosen) keys to manipulate on the keyboard (fire, up, down, left, right, battery etc.).

Missile Control is a game of skill, requiring fast reactions and dextrous fingers, and with tremendous appeal. This, however, is not all, for the best aspect of this game is losing! The graphics which appear on the screen at the end of the game are of such a fine quality they rival the game itself.

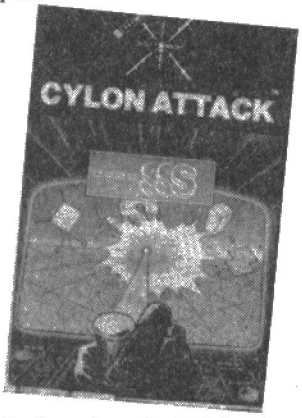
The BBC is one of the micros best served by the new 3D games. One such is **3 Deep Space** which uses the same green/red stereoscopic system as the 'Creature of the Black Lagoon' and other such movies. The graphics are however simple line drawings, with no blocks of colour or background. Nevertheless the effect is interesting and complemented by a good space game, in which alien objects (of a not too interesting shape in the first sequence) are destroyed with either 3 devastating missiles or by air attack. In the second phase the aliens look suspiciously like insects.

Bugbytes **Oblivion** is an unusual and refreshing adaptation of Space Invaders. Imagine that game, but with the players gun able to move up into the air as well as from side to side, and the enemy ships wheeling in from all angles, and you will have a good idea of Oblivion (the source of the title escapes me).

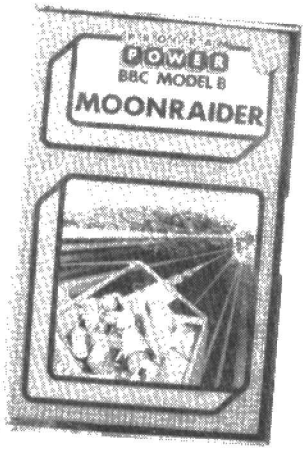
Also from Bugbyte is **Sea Lord.** This is not of the quality of Oblivion. While sailing around in your minisub you stray into the territory of a malevolent Sea Lord who's voice, according to the program blurb, booms out over your radio (not so, bugbytes technology stretches only as far as a textual representation). The 'power crazed dictator' then sends his mini-subs, or if you



destroy them submarines, and finally 'fighters'(!) to attack you. This is one of those games in which velocity is determined by rotating the craft, which does not promote accuracy of fire. However it is still all too easy to destroy the enemy, and this game is unlikely to have any lasting appeal.



Cyclon Attack (A&D Software) and **Lunar Rescue** (Alligata) are similar in that both require launching and docking from a mother ship. The former relies for fun on the gratuitous destruction of the skull shaped enemy ships, while the aim of the latter is to rescue fellow astronauts marooned on the lunar surface (perhaps a more worthy aim?). Cyclon Attack is a cockpit eye view game, with instruments indicating fuel, shield status, mother ship status, weapons left etc., and a gun aiming device which changes shape as the enemy draw into it. An excellent game but not worthy of the description '3D' which is found on the cover.



Last of the BBC games reviewed is **Moon Raider,** from Program Power. This game turns the tables and the player becomes the attacker looking for 'alien' bases on the moon (can there be any other kind?). Again you have to watch your fuel level while searching for enemy radar, rockets, ack-ack batteries (these aliens aren't too sophisticated) and bouncing space mines. Fortunately Fuel Transports are available to replenish the supplies. Three levels of difficulty are available.

Educational and utility packs now available for the BBC include a series of language tutors

(**The German Master** and **The French Mistress** – the double entendre is no doubt intended – from Kosmos Software), **The Generators** (a graphics and text generator by Quicksilva), and **Flexibase** which is a home filing system by Alligata.

'The Generator' allows the user to create graphics and text screens in Mode 7 of the BBC, and the beauty of this system is that the maximum of 12 screens (1K per screen) can be loaded to memory, saved and recalled at any time to be used within the users own programs (using a special CHAIN routine). The screens are created using the function keys to denote seven colours for a selection of 6 graphics blocks. It is very easy to create the screen backgrounds: observing their use in a particular program (especially one with moving sprites) would necessitate further research however. On the face of it this system is an excellent buy, particularly for children who will be able to indulge their creative abilities.

French Mistress and German Master are teaching programs to learn and test vocabulary, limited grammar, and gender (in the case of the French Mistress). As an example, on the French Mistress there is a series of four files each with eight lessons varying from vocabulary through to tenses, adverbs and premonitions, verb infinitives, conjunctions etc. The user enters a string in answer to each question. New lessons can be created by the teacher.

The Flexibase database sets up a maximum of 32 files each with a capacity for 254 records separated by field name. Features include sort (alpha only) and a search option which will match any string in any field, or in one field, less than or greater than match string, or a combination of any of the above using <and> and <or> logic operations.

ORIC

Oric software is not always of the finest quality, and is available in scant quantity; certainly that is the impression gained from this month's offering of the software houses.

The abysmal **Moonster** from Quark Data, while not typical, is the worst of a mediocre bunch. The most exacting element of this game is to discover which keys move the sprites and fire their intergalactic death rays. The answer must be achieved by trial and error, and I will not spoil anybody's pleasure by revealing the solution.

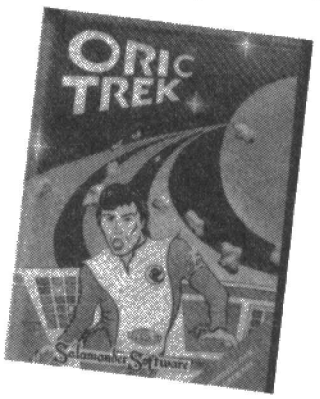
What remains is a poor man's space invaders, probably impossible to lose blindfolded, followed by an equally facile (monchrome) maze sequence in which an ungainly mess on the screen is manoeuvred past the evil ghouls to capture the Ruby in the cave(!). The response of the sprite to the keys is no less sluggish than a sloth in a state of advanced rigour, but despite this the ghouls never catch up and the 'game' is over in seconds. One thing only is less inspiring than Moonster, and that is its packaging; this will have the positive effect that only the hardened games junky will be attracted to, and disappointed by a pitiful product.

By contrast, **Oricmunch** is a professional quality Pacman variant from Tansoft, with beautiful sound effects and colour, likely to provide hours of entertainment for those with no more pressing mental commitments. Nine levels of difficulty are available, and it is possible to start at 1 and work up to 9 during the game, or to begin at 9 and work up to many millions of points at one's leisure.

Adventure games, according to current prejudice in the distribution trade, are purchased largely by adults. Infants, it seems, prefer the immediate gratification of blasting asteroids from the sky, and I cannot say I blame them.

Hell's Temple (Kenema Associates) is a typical example of its kind, a product of the dark ages when compared to the new 'real-time' 3D adventure games now available. Hell's Temple boasts some 70 different monsters to battle with, but each one lies only in the imagination and not on the screen, and there are few occasions during the play when the user has control over his/her destiny.

Salamander have two new games for the Oric on the market this month. The first, **Oric Trek,**



is a clash between Captain Kirk's USS Enterprise and the massed forces of the Klingon starfleet. The screen is divided into two sections: a short range scan and a long range scan, plus status and damage control reports from the likes of Scotty, Sulu, Uhura and Chekov.

which to send Klingon warriors to an early grave. This is an excellent and well produced game, but, with the vital instruction sheet missing from the review copy (oh frustration!) its full potential could not be realised.

The second Salamander product is a games compendium featuring **Donkey Derby, Viper** and **Kingdom,** and a space docking game, **Space Station.** The player must dock his ship with only limited fuel available. Two perspectives are given of the ship on a split screen, and the ship is manoeuvered across three vectors. Just to complicate matters, stray remnants of a meteor storm are passing by the space station as you dock. Contact with these has unfortunate results for the astronaut.

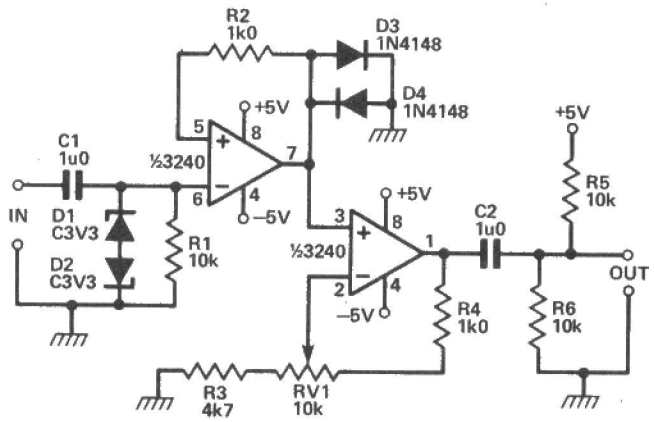
On the more serious side of life (what could be more serious than computer games?) is **Oricmon** (PSS). This is an extensive machine code monitor and disassembler. It permits hex codes to be entered, RUN and debugged. The full 6502 disassembler gives various memory displays and modification commands. A complete set of commands for debugging includes breakpoints, hex calculation and conversion commands.

Filestar (Kenema Associates), as its name suggests is a file handling system with sort, record search and line search, and 13 various editing commands. Only one file can be handled at any time, and is automatically deleted on creation of a new file. Address fields extend over just 16 characters maximum. The program is very fast – it is equal to professional quality in this respect – and leaves enough space in the 48K Oric RAM for quite extensive files. One bugbear is that the screen all too often goes right back to the menu if a mistake is made. Hopefully this is a tendency which could be overcome by someone more familiar with the system.

Photon torpedoes, hyperprobes, shield controls and warp drive are all useful little tools with

E&CM

CIRCUIT TIPS



A new occasional feature presenting readers' circuit hints, tips and ideas.

This month Ralph Lovelock describes a Pulse Recovery Circuit.

In the September issue of *E&CM* a useful cassette signal conditioner was described, using an operational amplifier connected to form a Schmitt trigger. The writer warned that the rise time was more than that from a normal digital ciruit, but was more successful in dealing with fast tape transfer due to the greater degree of hysteresis. The only great disadvantage for many constructors is that without an expensive cathode ray oscilloscope it cannot be reliably set up, or checked later; the use of miniature preset to adjust a positive feed-back would introduce a degree of 'wander', as might also the use of electrolytic capacitors with small, but significant leakage current. The unit described here is only slightly more expensive, and no larger physically, and uses the dual 3240 equivalent of the 3140 used in the conditioner described.

The circuit is shown above, and has the one great advantage that the functioning can be calculated, and then checked statically with little more than a normal test meter reading up to 5 volts. The two capacitors used are miniature Siemens polycarbonate with negligible leakage, a dual supply is used to obtain the stability of a central ground point not dependent upon a varying driven tap across a single-ended one, the two threshold levels are symetrical about ground, giving a greater and more stable degree of hysteresis, and the pulse rise-time is restricted to the amplitude of the pulse actually required.

The input voltage is restricted to a level safe for the 3240, C1 R1 locate the input symetrical about 0V and D1 D2 limit its amplitude to 7.9V peak to peak. D3 D4 limit the output from the Schmitt to 1.3V peak to peak, so that an input exceeding 1.5V p/p will suffice to trigger the circuit, while the input current can rise to 120mA RMS without damaging the circuit, or significantly spoiling the output wave-form.

The output is limited to values which will not damage TTL chips. The second amplifier has gain variable by R3 R4 VR1 such that the output will lie between 1.5V and 4.5V p/p. The output from the Schmitt is not quite symetrical about G because it is driven to SC current in each direction, and the output current positive to G is around twice that negative to G, but C2 R5 R6 move it to symetrical about the average value between 5V and G.

Providing the input voltage is kept well above 1.5V p/p it will be approximately symetrical for ± 0.65V so that the output will closely approach square, and the transition of Schmitt output should be about 120nS p/p. The feedback on the second amplifier will ensure that the generated wave is independent of load, and that the output will be similar in shape, with p/p transition between 120nS and 500nS depending on the setting of VR1. VR1 is a linear wire-wound component that will not suffer the wear and noise problems associated with carbon tracks.

The development model was assembled on a piece of plain (no tracks) Veroboard. The resistors and diodes were all mounted vertically to minimise strays and feed-back, and the 3240 was mounted in a socket to avoid danger of damage to the sensitive input sections.

E&CM

INTRODUCING 6502 AND Z80 VECTORS

It is a great pity that even now, after the explosion of home computing, many useful computing techniques remain intelligible only to a few initiates. One such device, more mythical than real to many programmers, is the use of vectors and the closely related subject of indirection. Adam Denning, in the first of two articles, sets out to demystify the mysterious.

At the most basic level, vectors are closely connected with bus routes – getting somewhere by an indirect route! Put simply, a vector is an entity that points to another, and naturally in computing terms these entities are more often than not memory locations.

It follows, therefore, that vectors are one way of implementing indirection, a strange word that means getting to one place via another. BBC Micro owners will immediately be reminded of that computer's 'indirection operators' (?, ! and $), and may be wondering exactly how these operators came to be given their 'indirection' prefix. This is one area in which the User Guide falls down, as it fails to explain indirection, the implication being that one would only require to use the operators if one understands indirection, which is rather arrogant.

Using the '?' indirection operator as an example, its common usage is as a unary operator, that is, only one operand/argument is specified. Thus,

```
PRINT ?24576        and   ?24576 = 255
```

are examples of unary operation. In fact, they are directly equivalent to

```
PRINT PEEK 24576        and   POKE 24576,255
```

From this, there seems to be no real reason why these operators should be described as indirection operators. However, the BBC Micro has a number of clever little tricks up its sleeve, and one of these is using ? and ! as binary operators; that is, with two operands. Unfortunately, the BBC Micro places a restriction on the user in that the leftmost operand in these cases must be a variable rather than a constant, but in practice this is no real drawback. It is just an annoying show of indirection! When being used in this way, the syntax is:

```
variable ? (or !) offset
```

where 'variable' is the base address under consideration and 'offset' is a number (which can be a constant, a variable or an expression) determining the actual address being examined or changed by the instruction. Thus, if Add = 24576, then

```
PRINT Add ? 35
```

would print out the contents of address (24576+35), which is 24611, and

```
Add ? 40 = 200
```

would place 200 into memory location 24616. The indirection is more obvious here.

To really get to grips with vectors, we have to descend to machine level and start talking in machine code terms. Apart from the Dragon and a very few other micros, the majority of computers under £500 use either the 6502 or the Z80, so it is sensible as well as convenient to discuss vectors in terms of these processors. In any machine-dependent parts of this article, the BBC Micro will be used as an example of a 6502 based machine and the Spectrum will take the part of the Z80 computer.

Although each processor impliments vectoring in slightly different ways, there are enough similarities for a 6502-only person to understand the Z80-specific parts and vice versa. There are also numerous examples of how to extend each processor's techniques to be applicable to the other, to show that the principles are truly universal.

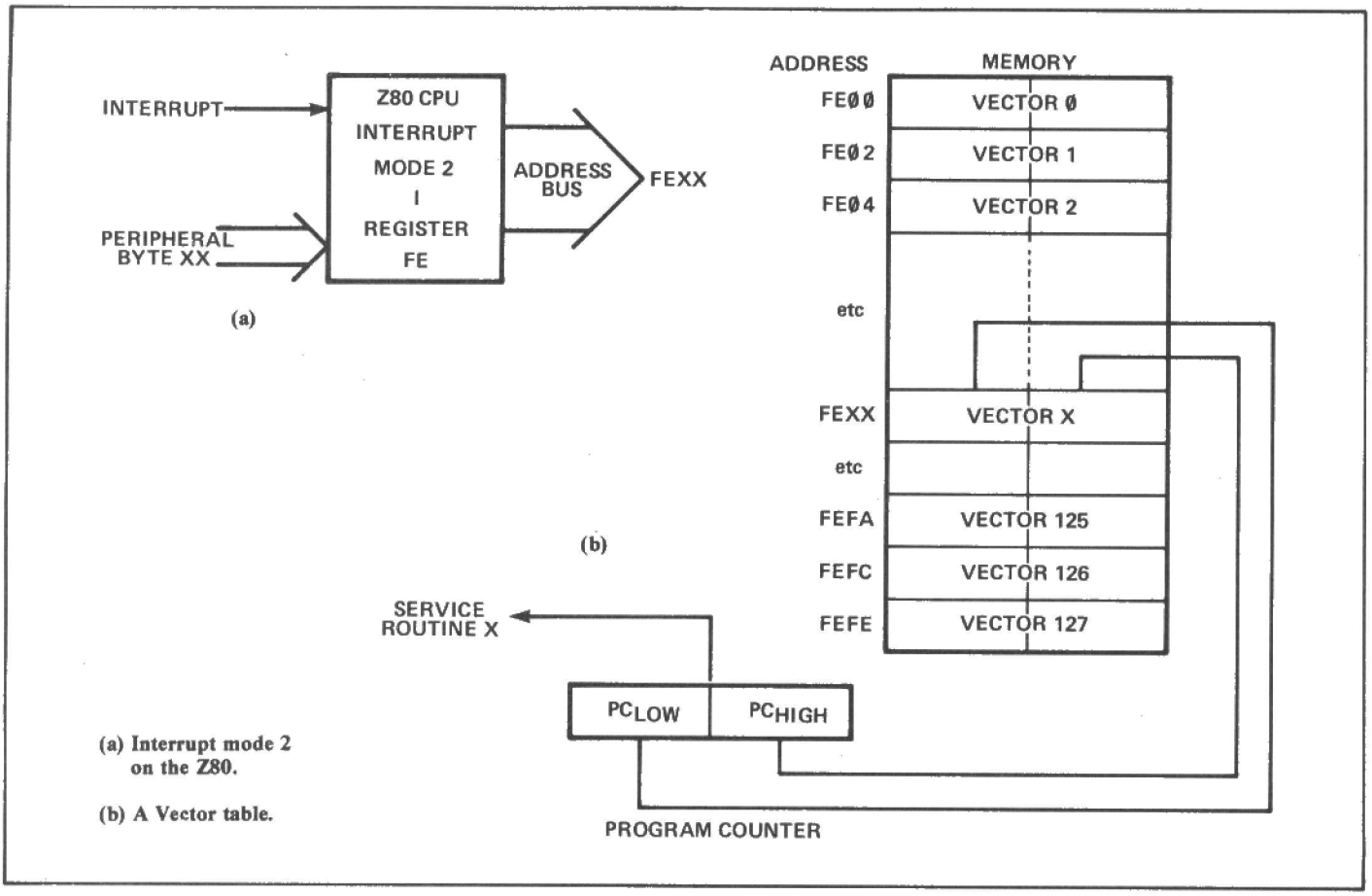
As previously stated, a vector is a pointer to somewhere else, and the need for this sort of utility arises in a number of instances at processor level. For example, the servicing of interrupts, the scanning of lines in a compiler or interpreter (or even in an adventure game) and data structures such as linked lists. In fact, some languages such as LISP, FORTH and BCPL revolve around vectors, the latter being based on the principle.

Z80 Interrupt Modes

Taking interrupts first. On the Z80 there are three different interrupt modes, invoked by assembly instructions IM 0, IM 1 and IM 2. Note that during the servicing of an interrupt on this processor only the program counter is saved (on the stack), so an interrupt service routine would have to preserve other registers either by multiple PUSH instructions, or if possible, the much faster EXX and EX AF,AF instructions.

In mode 0, the Z80 gets information from the peripheral that caused the interrupt (in fact, it could be any peripheral) telling it which RST instruction to execute: no vectors here. In mode 1 an interrupt always causes the processor to undertake an RST 38 instruction. This is the mode that the Spectrum uses, and RST 38 on this system is the keyboard scanning and frame counter updating routine. Again, no vectors in the real sense of the word. Now, Interrupt mode 2 is where things start to get interesting. There is a register deep within the Z80 called the I register, which deals specifically with interrupts. In mode 2, as in mode 0, the processor gets some information from the peripheral causing the interrupt, but this time, this information forms the low byte of a vector table address. As may have been guessed, the I register forms the high byte. As a vector has to occupy two bytes per address, it is a convention that the least most significant bit (i.e. BIT 0) provided by the peripheral is always reset (i.e. zero), so that the address formed from it and the I register is even. This naturally means that a vector table has to start at an even address, such as FE00 if this convention is adhered to.

So, suppose a table was set up in memory starting at FE00, the I register was loaded with FE, interrupt mode 2 was selected and an interrupt occurred. The vector table should consist of up to 128 different service routine addresses, occupying 2 bytes each. Then, on an interrupt, the processor gets a byte from the peripheral, combines it with the I register, and thus forms a 16 bit address. This address will of course be within the vector table, and provided the reset bit 0 convention was followed, this address will contain a valid vector. This vector is then loaded into the program counter (this having been previously saved on the stack automatically), so that the processor 'vectors' to a service routine. At the end of this routine, one would normally have either a RET or a RETI instruction, which returns the processor to wherever it was before being interrupted.



(a) Interrupt mode 2 on the Z80.

(b) A Vector table.

Vectoring on a Spectrum

Obviously, this technique is extremely useful, as one can then have a system that can react in 128 different ways to an interrupt. On the Spectrum, however, things are not quite so useful as the hardware imposes a number of restrictions upon the user. Not least the constant DMA'ing (Direct Memory Accessing) of the first 16K of RAM to form the screen display, and the fact that any system still relying on BASIC must also execute an RST 38 as well as any other user routine on every interrupt. In effect, all this means is that the vectors cannot be placed in memory between 16384 and 32768, which would seem to rule out 16K Spectrums. This is because vectoring from this area of memory causes screen interference.

However, all is not lost. By ignoring the reset bit 0 convention, it is useful to know that the default value for unused ports is FF on the Spectrum, so an interrupt in mode 2 with no peripherals attached (standard peripherals such as the printer and the microdrive cause no problems) will cause the processor to vector from an address formed from the contents of the I registers and FF.

So, rather than a vector table, there would be one address in memory at address 'I FF', and this would be vectored to on an interrupt. Thus, on a 16K Spectrum, it would have to contain a value under 40 hex, and on a 48K machine, this value has to be between 0 and 3F or between 80 and FF to avoid screen interference. This involves being rather devious on a 16K Spectrum, as values of below 40 would mean having the vector in ROM, so it is necessary to search through the ROM looking for two adjacent bytes that together form a RAM address, in order to write a routine to look after the interrupt. These two addresses must also start at an address XXFF, where XX is the contents of the I register.

Examination reveals a few useful addresses, a convenient example being 28FF, which would cause a vector to 7E5C, which is well within a 16K Spectrum's RAM. A 48K owner has more flexibility and can either use this vector, search through ROM for some more suitable one, or create his own in RAM above 8000 hex. To cause the Spectrum to jump to this routine whenever an interrupt occurs, the following short program will suffice:

```
DI          ; disable the interrupts whilst things are being altered
LD A,28     ; load the I register with 28 hex via the A register
LD I,A
IM 2        ; select the correct interrupt mode
EI          ; re-enable the interrupts (otherwise all this will be
RET         ; pointless!) and return to BASIC.
```

Of course, this routine must not be run until a routine actually exists at 7E5C, or the system will crash. On a 48K Spectrum, things can be made more general with an assembler by including a routine called say 'INT2ON' in any program on which vectored interrupts are to be used, with the actual service routine being called something like 'INTRPT'. Then, an assembler file such as this will allow the use of interrupts whenever you like:

```
DI
LF HL,INTRPT
LD (VECTOR),HL
LD A,FE ; or whatever you decide – it is the high byte of
   'VECTOR'
LD I,A
IM 2
EI
RET
```

In both cases, the way to set the interrupt handling back to normal on a Spectrum is with this little routine:

```
DI .;again, dispose the interrupts during alterations
LD A,3F ; the normal value for I in the Spectrum system
LD I,A
IM 1
EI
RET
```

Listing 1 shows a short program using this facility to provide an accurate digital clock on the Spectrum, that runs all the time, even during BASIC programs. Of course, routines that disable the interrupts, such as BEEP and LPRINT, will also stop the clock. Note that the entire assembly listing is in a format suitable for direct input to Hisoft's GENS assembler, which the author feels is the best assembler there is for the Spectrum. Using the same principles, i.e. preserving all the registers during an interrupt, and ensuring that RST 38 is executed all the time, a whole host of interrupt driven routines can be used on the Spectrum.

6502 Interrupt Modes

The 6502 processor has only one interrupt mode, and this is always vectored. There are however, three different kinds of interrupt: non-maskable (the Z80 also has this, and an NMI on the Z80 results in a jump to 0066), maskable (the same type as the ones discussed for the Z80), and the software interrupt invoked by the BRK assembly instruction.

The 6502 processor is so designed that the three last words in the memory map (i.e. FFFA, FFFC and FFFE) always contain the vectors for these three types of interrupt. Again, the design of the 6502 is such that these are always in ROM, as, unlike the Z80, 'switching on' a 6502 system does not start the system executing from location 0000.

With a 1.0 or higher operating system on the BBC Micro, one is perfectly able to 're-route' these interrupts and thus create a highly useful system, or, through inadvertent indirection, a horrible mess.

Calculated Vectors

To seemingly go off the track for a while and leave practical examples until next month, it is still necessary to discuss calculated vectors on both processors. This will leave us in a position to fully understand the extension of Spectrum BASIC and various BBC utilities that will be demonstrated in part two. So meanwhile . . .

Consider the machine code instruction that represents a jump to an absolute address: in Z80 this is JP XXXX and in 6502 JMP XXXX, where XXXX is the absolute address in both cases. There is obviously no vectoring here, but on both processors there is method by which one can cause a jump to an absolute address that may be calculated, thus increasing the power of vectoring by a large degree. On the Z80, the 16 bit registers HL, IX and IY can form an instruction of the type

JP (regpair)

where regpair is either HL,IX or IY. This instruction loads the program counter with the contents of regpair, which therefore causes a jump to that address. Thus, a very powerful routine can be implemented where different addresses can be jumped to under differing conditions. For instance, examine the code below:

```
LD A,(PARSWD)
LD C,A
LD B,0 (this is zero)
LD HL,RTNTAB
ADD HL,BC
JP (HL)
```

Here, the contents of PARSWD are loaded into register A and then added to the base address of a table of routines, RTNTAB. This result is calculated in HL and thus the last instruction causes a jump to the relevant routine. A typical application for this sort of thing would be in text parsers, used in language compilers and interpreters, not to mention adventure games! The 6502 does not have this sort of jump, so before discussing its particular vectored jump, below is a method that serves to simulate the Z80 implementation:

```
LDA Addlow
STA Lowadd
LDA Addhigh
STA Highadd
JMP (Lowadd)
```

Where Addlow and Addhigh are the low and high bytes of the calculated address and Lowadd/Highadd is the vector itself.

As can be seen from the last instruction above, the 6502's vectored jump is somewhat different from the Z80's in that rather than jumping to the contents of a register it jumps to the contents of two adjacent bytes. Again, simulation of this idea on the Z80 is very simple:

```
LD HL,(Addlow)
JP (HL)
```

So, by indirect routes (that phrase again!) one can very simply achieve the desired result on either processor, but it is the author's opinion that the Z80 offers far more powerful facilities.

E&CM

Listing One – Digital Clock

```
  10 INPUT "HOURS? ",H,"MINUTES?
 ",M,"SECONDS? ",S
  20 POKE 65242,H: POKE 65241,IN
T (M/10): POKE 65240,M-10*PEEK 6
5241: POKE 65239,INT (S/10): POK
E 65238,S-10*PEEK 65239: POKE 65
237,0
  30 RANDOMIZE USR 65211
```

Assembly Listing


```
10  ;CLOCK ON :  USR 65211
20
30  ;CLOCK OFF:  USR 65226
40  *L-
50  VECTOR  EQU   #FEFF
60          ORG   #FE00
70  START   DI
80          RST   #38
90          PUSH  HL
100         PUSH  DE
110         PUSH  BC
120         PUSH  AF
130         LD    A,(CLKCNT)
140         INC   A
150         LD    (CLKCNT),A
160         CP    50
170         CALL  Z,NEWCLK
180         POP   AF
190         POP   BC
200         POP   DE
210         POP   HL
220         EI
230         RET
240 NEWCLK  XOR   A
250         LD    HL,CLKCNT
260         LD    (HL),A
270         INC   HL
280         INC   (HL)
290         LD    A,#0A
300         CP    (HL)
310         JR    NZ,PRTCLK
320         LD    (HL),#00
330         INC   HL
340         INC   (HL)
350         LD    A,#06
360         CP    (HL)
370         JR    NZ,PRTCLK
380         LD    (HL),#00
390         INC   HL
400         INC   (HL)
410         LD    A,#0A
420         CP    (HL)
430         JR    NZ,PRTCLK
440         LD    (HL),#00
450         INC   HL
460         INC   (HL)
470         LD    A,#06
480         CP    (HL)
490         JR    NZ,PRTCLK
500         LD    (HL),#00
510         INC   HL
520         INC   (HL)
530         LD    A,#18
540         CP    (HL)
550         JR    NZ,PRTCLK
560         LD    (HL),#00
570 PRTCLK  LD    HL,#50F8
580         LD    (PRTPOS),HL
590         LD    B,#02
600         LD    A,(CLKVAR+4)
610         PUSH  AF
620         CP    #14
630         JR    NC,CLKPRT
640         DEC   B
650         CP    #0A
660         JR    NC,CLKPRT
670         DEC   B
680 CLKPRT  LD    A,B
690         ADD   A,#30
700         PUSH  BC
710         CALL  IMITAT
720         POP   BC
730         LD    A,B
740         ADD   A,A
750         ADD   A,A
760         ADD   A,B
770         ADD   A,A
780         LD    B,A
790         POP   AF
800         SUB   B
810         ADD   A,#30
820         CALL  IMITAT
830         LD    A,#3A
840         CALL  IMITAT
850         LD    A,(CLKVAR+3)
860         ADD   A,#30
870         CALL  IMITAT
880         LD    A,(CLKVAR+2)
890         ADD   A,#30
900         CALL  IMITAT
910         LD    A,#3A
920         CALL  IMITAT
930         LD    A,(CLKVAR+1)
940         ADD   A,#30
950         CALL  IMITAT
960         LD    A,(CLKVAR)
970         ADD   A,#30
980 IMITAT  SUB   32
990         LD    L,A
1000        LD    H,0
1010        ADD   HL,HL
1020        ADD   HL,HL
1030        ADD   HL,HL
1040        LD    DE,#3D00
1050        ADD   HL,DE
1060        EX    DE,HL
1070        LD    HL,(PRTPOS)
1080        PUSH  HL
1090        LD    B,#08
1100 RSTTEN LD    A,(DE)
1110        LD    (HL),A
1120        INC   DE
1130        INC   H
1140        DJNZ  RSTTEN
1150        POP   HL
1160        INC   HL
1170        LD    (PRTPOS),HL
1180        RET
1190 INT2ON LD    HL,START
1200        LD    (VECTOR),HL
1210        DI
1220        LD    A,#FE
1230        LD    I,A
1240        IM    2
1250        EI
1260        RET
1270 INTOFF DI
1280        LD    A,#3F
1290        LD    I,A
1300        IM    1
1310        EI
1320        RET
1330 PRTPOS DEFS  2
1340 CLKCNT DEFB  0
1350 CLKVAR DEFS  5
```

Part 2

HI-RES COMPUTER THE A/D BOARD

Paul Izod and Alan Stirling continue their description of the A/D board of our 6809 based computer.

The full circuit diagram of the Hi-Res Computer's A/D board was given in our October issue. This revealed that the design adopted a 'hardware intensive' approach to the task of A/D conversion in order to reduce the software overhead when using the system.

In view of the large number of components featured in the project it is strongly recommended that the custom PCB designed for the assembly is used to construct the A/D system.

Before Building

For many applications, some of the analogue components won't be necessary. As an example, if only one channel is to be measured, then neither of the analogue multiplexors will be required. If that channel is single ended, then the AD524 instrumentation amplifier may be omitted too. The sample and hold circuit is only of value if the input signal might change by more than, say, ½ LSB during conversion.

If a component isn't to be fitted then bypass it with a link from input to output. Each DAC has 2 octal registers used as input latches. If either DAC0 or DAC1 isn't wanted then neither will the latches.

There are a number of offset and gain adjustment trimpots on this board, and as their adjustment range (% of FSR) exceeds the uncalibrated error of the 12 bit converters, it is better not to fit the trimpots and associated resistors unless a DVM is available to set them up. Ideally, the DVM should have at least 5 digits, resolve to 100μV on the 10V range and be accurate to 0.01% or better. Remember that the calibrated accuracy will depend upon the DVM. In any case, to help with any faultfinding that may be necessary, don't fit R1 to R6 and R14 to R29 until after initial testing.

Links

The board was designed for a processor clock frequency of 1MHz. Q_B of D29 is available if the E clock is 2MHz. The track to Q_A must be cut if Q_B is used. (IC11).

Decide whether to use the ADC80 or the ZN427, and wire the appropriate link, just below IC7. If your system uses address lines A16 and A17, cut the track from E to F and wire a link from F to A, B, C or D as required. (Beneath IC6).

A link from L to either M or N will cause an interrupt when all channels have been converted. If no routine exists to clear the source of interrupt (by reading the board), then omit this link. L-M will cause IRQ; L-N for NMI.

Lastly, it is necessary to wire links according to the desired voltage ranges of the DACs and the ADC80. See **Table 1.**

Construction

It is strongly recommended that good quality sockets be used for all of the analogue components (except LF398). If a 32 pin socket for the ADC80 proves difficult to find, then use 2 lengths of single-in-line socket strip. Whether or not sockets are used for the digital ICs is more a matter of personal preference.

Having decided on the links required, fit these together with the passive components (not R1 to R6 and R14 to R29), PCB connector blocks and IC sockets.

Last of all, add the ICs. Be sure that the analogue supplies are of the correct voltage and polarity before inserting the completed board and powering up.

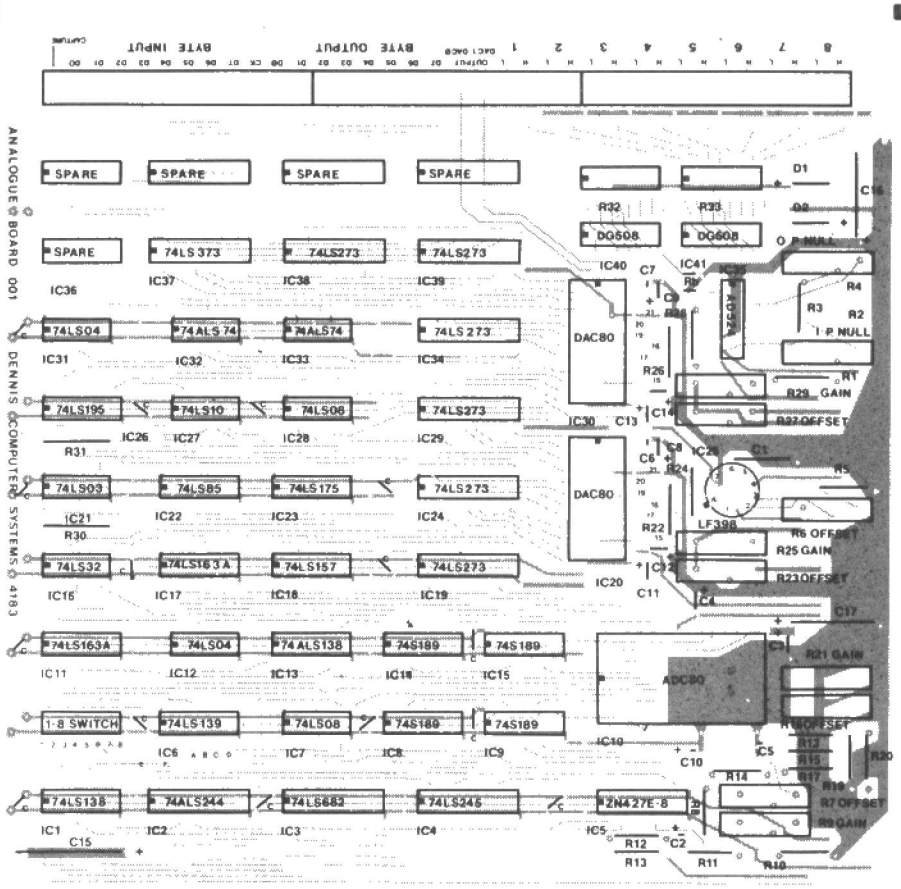


Figure 1. Overlay of the A/D board. Note only the topside foil pattern is shown.

ADC80 INPUT SCALING

INPUT RANGE	OUTPUT CODE	PIN 12 TO	PIN 14 TO	INPUT TO
±10V	COB	11	INPUT	14
±5V	COB	11	—	13
±2.5V	COB	11	11	13
0 TO +5V	CSB	15	11	13
0 TO +10V	CSB	15	—	13

DAC80 OUTPUT VOLTAGES

OUTPUT RANGE	INPUT CODE	PIN 15 TO	PIN 17 TO	PIN 19 TO
±10V	COB	19	20	15
±5V	COB	18	20	–
±2.5V	COB	18	20	20
0 TO +10V	CSB	18	21	–
0 TO +5V	CSB	18	21	20

COB: COMPLEMENTARY OFFSET BINARY
CSB: COMPLEMENTARY STRAIGHT BINARY

Table 1. Wire link connections.

Initial Testing

Configure the board as Board 1 (links A to F and 1 out of 8 switch).

Byte Output

Use a voltmeter or logic probe to check that all bits of the 'Byte Output' register have been reset. Using the Monitor, write FF to location F205. Verify that all bits are set.

Byte Input

With a 74LS373 for IC37 and the clock input floating, location F200 should read FF. Ground each input in turn and read F200 to ensure that they go low.

DAC0

These converters have complementary binary inputs. After a system reset, the outputs of DAC0 and DAC1 will be at their most positive value. The actual voltage will depend upon the links connected; check that the voltage is correct.

Write FF to F201 and F202. The output voltage should be the most negative. Use other binary values to exercise the converter more thoroughly.

DAC1

As DAC0 but use locations F203 (MSByte) and F204 (LSByte).

ADC80/ZN427

Ensure that there is a continuous signal path from the input connector block to the A-D Converter. Connect channel 1 low input to ground and High input either to an external voltage source which is variable over the ADC input range or to the output of DAC0 or DAC1. Set up the board to continuously scan channel 1, with internal trigger, by writing 40 to the configuration register F200. Trigger the board by writing to F207 (data unimportant). Channel 1 should now be continuously digitised. Read locations F280 (ADC80 only) and F281 (both) to see the digitised data. Vary the input voltage and observe the value change (re-read F280 and F281 each time). The 4 MSBs of F280 will be 0 if IC8 is fitted and F if not.

The other channels may be checked similarly.

Calibration

It is suggested that if the Sample and Hold is fitted, then this should be adjusted first, to avoid affecting the A to D converter.

LF398

The offset should be adjusted with the input of the device grounded and whilst continuously switching between sample and hold.

Remove the AD524 from its socket and ground the input of the LF398. Set up the configuration register for repetitive conversion and trigger the board.

Adjust R6 until the output of the LF398 is zero. Remove the short from the input and replace the AD524.

AD524

The input offset voltage of the AD524J is 250μV max (at 25°C and ±15V), and output offset voltage is 1mV max, so the null adjustments will probably not be required. Note however, if R2 and R4 and fitted, they will have to be set up. R2 will null the I/P offset and R4 the O/P.

ADC80

Set up the configuration register for repeated conversion of channel 1, and trigger the board. Connect channel 1 to a variable voltage source of known voltage. This may take the form of a power supply, potential divider and 5 digit DVM.

Set the analogue input voltage to −ve full scale + 1LSB (−9.9951V for ±10V), see **Table 2.** Adjust the offset trimpot R16 until the digital output is 1111 1111 1110 (LSB). Next set the input voltage to the +ve full scale −2LSBs (9.9902V for ±10V). Adjust the gain trimpot R21 for a digital output of 0000 0000 0001 (LSB).

Please note that the digital outputs are referred to the ADC80 itself. If 74S189 RAMs are fitted and the Monitor used to observe the outputs, the values shown must be inverted. E.g. −ve full scale would be adjusted for 0000 0000 0001.

ZN427

The printed layout establishes the input range to be bipolar, the component values determine the voltage range.

Set up the configuration register and trigger the board as for ADC80.

1. Apply a voltage −(full scale – ½LSB) to the input and adjust the offset R7 until the LSB just flickers between 0 and 1 with all other bits 0.
2. Now apply +(full scale – 1½ LSBs) and adjust the gain trimpot R9 until the LSB flickers 0-1 with all other bits 1. Repeat step 1.

Note that again, the logic levels are referred to the ZN427. The flicker on the LSB is really quite obvious, and may be seen using an oscilloscope or reading channel 1 in a program loop.

For an input range of ±10V

−(full scale + ½LSB) = −9.961V
+(full scale −1½ LSB) = 9.883V

These voltages may be generated using either DAC0 or DAV1 set to an O/P range of ±10V. 1111 1111 1011 (LSB) should give approx. −9.961V and 0000 0001 0111 (LSB) should give approx. 9.883V.

DAC80

Allow a warm-up period, after power on, of at least 30 seconds. Write all ones to the input registers. This should produce the most negative O/P voltage. Vary the offset adjustment (R23 for DAC0 and R27 for DAC1) until this voltage is produced. See **Table 3.** Next write all zeros to the input registers. Adjust the gain trimpot (R25 for DAC0 and R29 for DAC1) for positive full scale voltage.

DIGITAL INPUT	ANALOGUE OUTPUT			
	VOLTAGE*		CURRENT	
MSB LSB	0 TO +10V	±10V	0 TO −2mA	±1mA
0000000000000	+9.9976V	+9.9951V	−1.9995mA	−0.9995mA
0111111111111	+5.0000V	0.0000V	−1.0000mA	0.0000mA
1000000000000	+4.9976V	−0.0049V	−0.9995mA	+0.0005mA
1111111111111	0.0000V	−10.0000V	0.0000mA	+1.0000mA
ONE LSB	2.44mV	4.88mV	0.488uA	0.488uA

*TO OBTAIN VALUES FOR OTHER BINARY RANGES:
0 TO +5V RANGE: DIVIDE 0 TO +10V RANGE VALUES BY 2
±5V RANGE: DIVIDE ±10V RANGE VALUES BY 2
±2.5V RANGE: DIVIDE ±10V RANGE VALUES BY 4

Table 3. DAC 80 calibration details.

Corrections And Additions

Some minor errors appeared in the text and circuit diagram published last month. Also, the layout of the printed board has necessitated some changes.

The letter y was frequently printed instead of μ (micro) – definitely microseconds and not years!

The circuitry around D1 (74LS139) should have been as **Fig. 2.**

The Signal Reset is no longer buffered by D4 and D5 (74LS03) but by a spare element in D9. Similarly, address line A3 is now buffered by D9. These changes are shown in **Fig. 3.**

Please note the new pin designations, caused by the printed layout.

The interrupt function, generated by gating $A = B^{n+1}$ with Q_D using D27, has been changed. The new circuit causes the interrupt to be latched until the board is read. The revised section of the circuit is shown in **Fig. 4.**

A connection should have been shown from pin 11 of the Read Input Byte register (D51) to the connector block. This is the Clock input. D51 may be either a 74LS373 or a 74LS374.

DAC0 and DAC1 were meant to have been shown configured for an output voltage range of ±10V but inadvertently, pins 15, 18

Table 2. ADC 80 calibration table.

BINARY (BIN) OUTPUT	INPUT VOLTAGE RANGE AND LSB VALUES					
ANALOGUE INPUT VOLTAGE RANGE	DEFINED AS:	±10V	+5V	±2.5V	0 TO +10V	0 TO +5V
CODE DESIGNATION		COB OR CTC*	COB OR CTC*	COB OR CTC*	CSB**	CSB**
ONE LEAST SIGNIFICANT BIT (LSB)	$\frac{FSR}{2^n}$	$\frac{20V}{2^n}$	$\frac{10V}{2^n}$	$\frac{5V}{2^n}$	$\frac{10V}{2^n}$	$\frac{5V}{2^n}$
	n = 8	78.13mV	39.06mV	19.53mV	39.06mV	19.53mV
	n = 10	19.53mV	9.77mV	4.88mV	9.77mV	4.88mV
	n = 12	4.88mV	2.44mV	1.22mV	2.44mV	1.22mV
TRANSITION VALUES MSB LSB						
000...000***	+FULL SCALE	+10V −3/2LSB	+5V −3/2LSB	+2.5V −3/2LSB	+10V −3/2LSB	+5V −3/2LSB
011...111	MID SCALE	0	0	0	+5V	+2.5V
111...110	−FULL SCALE	−10V +½LSB	−5V +½LSB	−2.5V +½LSB	0 +½LSB	0 +½LSB

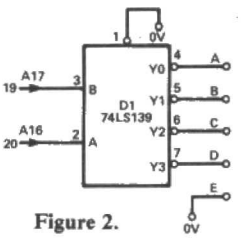


Figure 2.

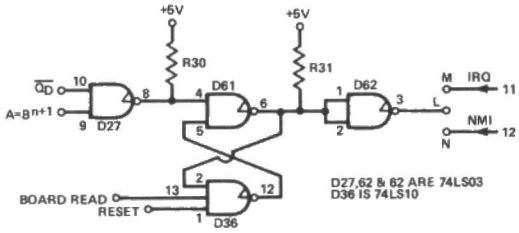


Figure 4.

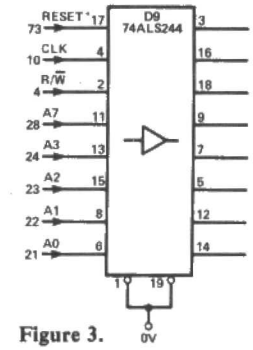


Figure 3.

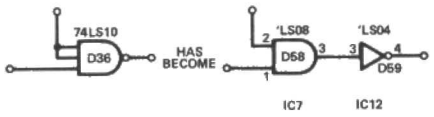


Figure 5.

Please note that in all figures the arrows shown on I/O lines signify a connection to the edge connector.

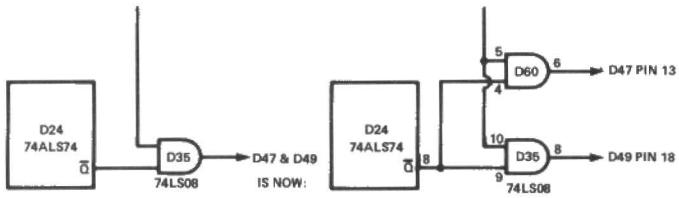


Figure 6.

and 19 were connected together and to R22. Only pins 15 and 19 should have been connected. R22 must be tied to the junction of pins 17 and 20. On the printed board, all of the output ranges can be selected.

Both inputs of the AD524 instrumentation amplifier must have a DC return path for their bias currents. Problems are unlikely to arise with single ended inputs which are ground referenced and DC coupled to the source, but provision has been made on the printed board to connect a resistor from each input to ground when measuring differential signals which aren't referenced to the analogue board ground.

To protect the analogue multiplexors from inputs greater than the supply voltage, and to allow the inputs to remain connected when the system is switched off, diodes have been added in series with the power supply connections and 2k7 resistors placed before each input.

There are 2 minor changes to avoid excessively long tracks as detailed in **Fig. 6.**

To avoid the analogue ground sharing the digital ground on the backplane, new pins have been designated for the analogue power supply.

+15V pin 70
−15V pin 71
A0V pin 72.

Please note carefully that these pins were used by the CPU board for quite different (and now redundant) purposes and the tracks to the edge connector on the CPU board must be cut. The existing ±12V pins may be satisfactory for 8 bit performance.

Finally, a fourth data RAM (D61) has been added to the circuit. This avoids bits D4-D7 being undefined when the most significant byte of the ADC80 is read. Previously they floated high, now they are read as zero. This will simplify the software when the board is accessed by a high level language not having bit manipulation facilities. D61 may be omitted.

PARTS LIST

Semiconductors

IC1	74LS138 (D3)
IC2	74ALS244 (D9)
IC3	74LS682 (D2)
IC4	74LS245 (D6)
IC5	ZN427E-8 (D48)
IC6	74LS139 (D1, D28)
IC7	74LS08 (D46, D33, D34, D58)
IC8	74S189* (D61)
IC9	74S189* (D38)
IC10	ADC80** (D49)
IC11	74LS163A (D29)
IC12	74LS04 (D8, D26, D45, D47, D59)
IC13	74ALS138 (D20)
IC14	74LS189* (D39)
IC15	74S189* (D40)
IC16	74LS32 (D7, D15, D16, D18)
IC17	74LS163A (D13)
IC18	74LS157 (D3)
IC19	74LS273 (D10)
IC20	DAC80** (D54)
IC21	74LS03 (D27, D61, D62)
IC22	74LS85 (D11)
IC23	74LS175 (D14)
IC24	74LS273 (D52)
IC25	LF398 (D42)
IC26	74LS195 (D30)
IC27	74LS10 (D12, D32, D36)
IC28	74LS08 (D17, D19, D35, D60)
IC29	74LS273 (D53)
IC30	DAC80** (D57)
IC31	74LS04/74LS14 (D25, D31)
IC32	74ALS74 (D21, D22)
IC33	74ALS74 (D23, D24)
IC34	74LS273 (D55)
IC35	AD524 (D41)
IC36	SPARE
IC37	74LS373/74LS374 (D51)
IC38	74LS273 (D50)
IC39	74LS273 (D56)
IC40 & IC41	DG508 (D43 & D44)

* 74LS189 is preferable but not easy to find. Both have inverted outputs. Use the 74LS289 for non-inverted outputs (for ZN427).

** **For ADC80:** ADC80AGZ-12, ADC80AG-12, AD ADC80Z-12, or AD ADC80-12
For DAC80: DAC80-CBI-V, AD DAC80-CBI-V, DAC1280, DAC1280A, DAC800-CBI-V or DAC800P-CBI-V

The suffix Z models of ADC80 are wide supply range and must be used for ±12V operation.

Resistors (all 0.4W metal film unless otherwise marked)

R1, 3	10k
R2, 4, 16, 21, 23, 25, 27, 29	10k cermet trimpot
R5	24k
R6	1K cermet trimpot
R7	10K cermet trimpot (5K)*
R8	27K (13K)*
R9	5k cermet trimpot
R10	8k2 (13k)*
R11	8k2 (7k5)*
R12	390R
R13	180k (12V supplies) 220k (15V supplies)
R14, 15	180k
R17	22k (28k if ADC80Z)
R18, 20	270k
R19	6k8 (9k1 if ADC80Z)
R22, 26	3M9 carbon
R24, 28	10M carbon
R30, 31	10k
R32, 33	2k7 x 8 D.I.L.

* Use values in brackets for input range of ±5V on ZN427.

Capacitors

C1	1500p polyester, teflon or polypropylene (Chold)
C2, 3, 4, 8-14	1μF 35V tantalum
C5-7	10n monolithic ceramic
C15	47μF 25V solid aluminium
C16, 17	10u 25V solid aluminium

Semiconductors

D1	IN4148
D2	IN4148

Sockets

14 pin x 9
16 pin x 16
18 pin x 1
20 pin x 10
24 pin x 2
32 pin x 1 (may be made using S.I.L. socket strip)
Single pole 8 way D.I.L. switch
12 way PCB connector blocks x 3

LETTERS TO THE EDITOR

Send your programming or construction hints, comments, queries, or complaints! to *The Editor, E&CM, Scriptor Court, 155 Farringdon Road, London EC1R 3AD.* £10 is paid for the star letter of the month.

★ £10 Star Letter

Sir,

The Sinclair Spectrum needs a TRACE command to print program line numbers, variables, etc., as each one is executed by the computer; a useful aid to debugging.

My method makes use of the printer, as I find that printing TRACE information on the screen leads to confusion. By setting a 'switch' at the start of the program you can turn the TRACE on or off.

Here is the system applied to the program printed on page 38 of the Spectrum manual.

```
    1 LET TRACE=1: REM The switch
'on' to use trace. To turn off
the trace make this line
      LET TRACE=0
  100 LET x=10: IF trace THEN PRI
NT #3;x; REM This will give the
value of the variable.
  110 GO SUB 500
  120 IF TRACE THEN PRINT #3;"L12
0"; PRINT s
  130 LET x=x+4
  140 GO SUB 500
  150 IF TRACE THEN PRINT #3;"L15
0"; PRINT s
  160 LET x=x+2
  170 GO SUB 500
  180 IF TRACE THEN PRINT #3;"L18
0"; PRINT s
  190 STOP
  500 IF TRACE THEN PRINT #3;"L50
0"; LET s=0:      REM The 'L' to
indicate line number rather than
variable value.
  510 FOR y=1 TO x
  520 LET s=s+y
  530 NEXT y
  540 RETURN
```

Yours,
L. V. Philips
Leicester

More Spectrum Hints

Sir,

Thank you for the Spectrum Interface project in May 1983 issue of *E&CM.* There was a machine code program given for LList and LPrint: it did not cover the COPY command.

I have just purchased 'Masterfile' – a filing program that uses COPY rather than LPRINT when producing printer output; and I have therefore, written a short BASIC subroutine to emulate COPY on the Spectrum. It occurred to me that other readers might be interested in using it not least because the Spectrum manual fails to cover the use of SCREEN$ properly; and this is essential to such a program. I enclose a listing of the routine, and can supply a copy on tape if required.

```
8000 REM COPYRIGHT S.SPRINGETT 1983
8004 REM EACH LINE
8005 FOR L=0 TO 21
8010 REM CHECK LENGTH OF PRINTING REQUIRED
8015 FOR C=31 TO 0 STEP -1
8020 IF  SCREEN$ (L,C) <> " " THEN  GO TO 8040
8030 NEXT C
8035 REM PRINT THE LINE
8040 FOR N=0 TO C
8050 OUT 65407, CODE  SCREEN$ (L,N)
8060 NEXT N
8065 REM  LINE FEED AND CARRIAGE RETURN
8070 OUT 65407,13
8080 OUT 65407,10
8090 NEXT L
8100 RETURN
```

Yours faithfully,
S. P. Springett,
129 Guildfords,
Old Harlow,
Essex.

16 into 48 won't go?

Sir,

I am a 16K ZX Spectrum owner and I have attempted to upgrade it to 48K. I have however come across a snag, and I would appreciate your help. I can POKE and PEEK to the extra memory but when I use the addresses on the system variables I find that they do not respond in the correct way. For example when I type:

PRINT PEEK 23732 + 256*
PEEK 23733

I get a result of 32789; this neither corresponds to the old 16K value nor to the expected 48K value. I have replaced all the integrated circuits that came with the memories, i.e. IC23-26, but there is no change. Could you possibly tell me what is wrong.

B. Prajzner
Ashton-under-Lyne, Lancs.

P.S. I am a regular reader and in my view the contents each month are balanced and superb.

This is very odd and is causing us a good deal of brainache! You say that you are able to POKE and PEEK to the extra 32K of RAM but the all important question here is are the values PEEKed the same as the values POKEd there in the first place? If there is any inconsistency with the values then the new RAM must be at fault – either you have inserted the IC's incorrectly or something a little more fundamental is wrong. We suggest that you thoroughly check this as it seems the most likely cause.

The value returned from the P-RAMT system variable is very strange indeed. A close look at the ROM routines that set the value of this variable shows that it is formed from the RAM check initiated at switch-on. Thus, as the ROM cannot be at fault, this again points to faulty RAM chips or incorrect insertion.

If I were you I'd examine the Spectrum circuit diagram very closely to ensure that you are not doing something wrong. If you still cannot solve the problem, I suggest that you contact Sinclair.

Those Spectrum Vectors

Sir,

Your reader John de Rivaz (letters, August '83) is perfectly correct to complain that my article on the Spectrum printer interface stated, erroneously, that the Spectrum had no printer driver vector, but he would be better advised to make his complaint to the writers of the Sinclair handbook.

I wrote my machine code routine around Christmas of 1982 and, after tidying it up for publication, it passed out of my hands in February of 1983. Think on those dates. *All* the Spectrum handbook says is that an area of RAM is designated "Channel information", and that this area follows the Microdrive maps. The key to the Channels was not unlocked until Dr Ian Logan published the disassembly of the Sinclair ROM, which was *not* as early as February 1983. The vector in question is at 5CC5H (though that address will change when the microdrives are in use) and it is loaded with 09F4H at switch-on time. The address of a different printer routine would be placed there, most significant byte first.

I don't mind John de Rivaz getting some free publicity at my expense for his £1 BASIC listing which, I am sure, does exactly what he says it does, but if he really wants to be useful then he can look at the machine code listing on page 78 of July *E&CM.* There he will find, from 6F00H-6FB5H and 7116H-7132H, all the code he needs to dump the entire Spectrum screen to his EPSON 80 printer so he will be able to see his user-graphics as they really are, and not as mere numbers. He might like to incorporate something like it into his own BASIC routine.

I hope he finds the code up to standard.
Richard Sargent
Wantage, Oxon.

6805 Cross Assembler

Sir,

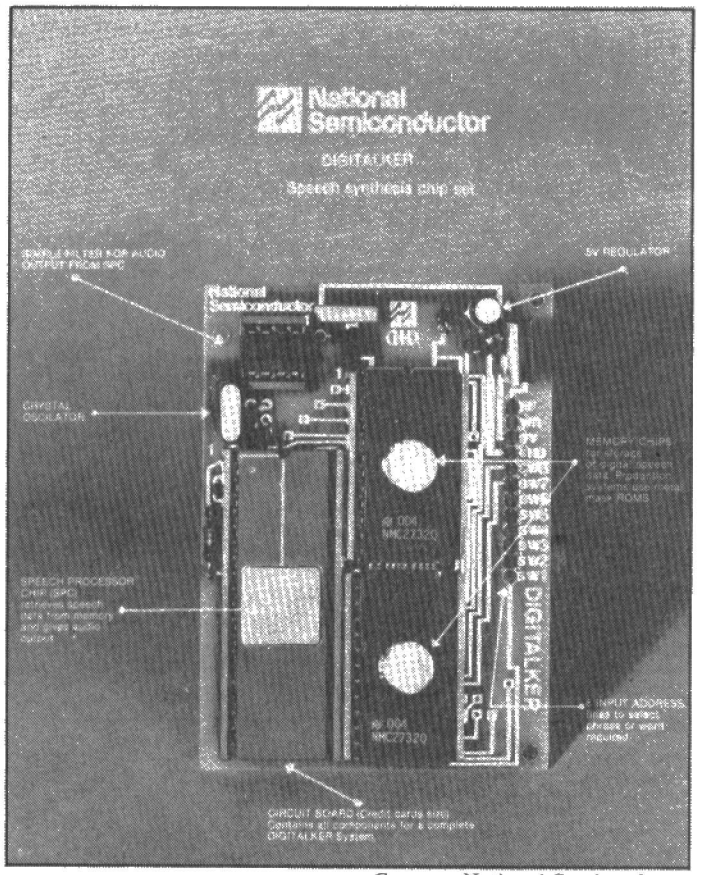
I'm really enjoying *E&CM* lately, but your recent articles on the 6805 really struck a chord. This chip intrigues and fascinates me: I got hold of Motorola's 'Design Five' evaluation kit, but quickly discovered the kit was not much use without programming the assembler facilities. You can buy a Motorola EXORset system for £10,000 and software etc. – a little bit over the top for most people.

So, can I encourage you, please do write your cross assembler, then the 6805 becomes a cheap viable microprocessor system.

On the off-chance, do you know where I can get hold of a 6805 cross assembler now, in BASIC or 6502 or 6800, 8080 or Z80 code?

Great mag. Cheers.
Eric Demmon
Walkern, Herts.

On the off chance, we do indeed know where you can get hold of a 6805 cross assembler. It will be appearing in E&CM early in the new year. Good luck with the 6805 system, and thanks for the compliment!



Courtesy National Semiconductor

DIGITAL SPEECH SYSTEMS

Ian Campbell concludes his examination of Speech Synthesis systems with a look at National's Digitalker and the techniques of Linear Predictive Coding.

Last month's article described the Allophone approach to speech synthesis adopted by General Instruments. Another approach is the one which has been adopted by National Semiconductor. In essence here there is no automatic breaking down of speech into its basic elements. What does happen is a sampling of an original speech waveform followed by digitisation, compression, adaptive delta modulation and then adjustment of the phase information in the digitised speech. By using this method phonemes, phoneme groups, words and even whole phrases can be recorded, placed in ROM and subsequently synthesised back into speech by a processor chip as required.

Let us now look at the whole process in a little detail. Firstly a high fidelity tape recording will be made by a prospective customer. This recording will be analysed by a rather complex computer program back at National Semiconductor. The really smart bit comes during the compression procedure which is applied to the voiced sounds after the voiced and unvoiced sounds are separated. It is a very important technique because it reduces the amount of memory space required to record the speech data. This is done by removing the excess or redundant data from the speech signal. There are four main procedures applied to bring that about, namely:-

1. Elimination of redundant pitch periods.
2. Adaptive delta modulation coding to minimise bandwidth and memory requirements.
3. Phase angle adjustments to create mirror image symmetry.
4. Replacing the low level portion of a pitch period with silence.

Figure 5 gives an illustration of what happens to the waveform of a typical voiced pitch period during its conversion. Part(a) shows the original waveform. Part(b) is the appearance after the phase delay is adjusted to create a symmetrical waveform about the centre of the pitch period. Next the low level beginning and ending quarters of the waveform are replaced with silence and last but not least, delta modulation is applied, and the final result is as in part(c). The result of all this is a compression factor of 4 to 1 on the original voice data.

As far as unvoiced sounds are concerned the situation is somewhat different. The reason for this is that unlike voiced sounds which are so speaker dependent and therefore rather variable, the unvoiced sounds have very few speaker defined characteristics. As a direct result of this just a small set of them can be used and compared with the speaker's.

Figure 6 shows a block diagram of NSC's MM54104 which is the SPC (Speech Processing Chip) that forms the basis of their 'Digitalker' (a minimum configuration for a Digitalker system will be seen in **Fig 7**).

Let's see now how the MM54104 deals with a little bit of speech synthesising. To get everything going the WR pin must go high, then the eight bit start address code will be loaded into the control word address register. The SPC will use this control address to fetch from ROM the control word for the first block of speech data. This very important word contains waveform information, repeat information and the address of the speech data. This address is loaded into the phoneme address register and is used to fetch the speech data that will be used to recreate the speech waveform. That is not all, because before synthesis can take place, the waveform data has to be decoded to provide information on such things as male or female speech, voiced or unvoiced sound and half period zeroed or not.

Of course all this movement of data must occur at the right time otherwise there would be an unholy mess up at the loudspeaker. This implies a system clock must lurk somewhere. It is in fact derived from a programmable frequency generator. Note that it is therefore a programmable clock! The control word allows for variations in the frequency of the clock which in turn allows the SPC to add a rising and falling in pitch to speech sounds and syllables.

This is the way the synthesiser makes the speech more realistic. In fact just as the frequency of the clock is determined by a control word at the beginning of the speech block so is the gain of the programmable gain amplifier. This further enhances the realism of the final speech pattern.

At the end of a waveform or speech block the SPC may repeat the sequence. Usually 3 to 4 times for voiced sounds and about 7 or 8 times for unvoiced ones. Having made the correct number of repeats the SPC moves on to the next speech block sequence. This operation continues until the SPC has carried out all the control words associated with the original start address code.

As far as the recreation of the waveform in **Fig 5(a)** is concerned, the SPC would deal with it as follows. It would start with a period of silence (in this case no speech data is required). Next there will be a quarter period of adaptive delta modulation generated speech, then the same speech data fetched in reverse and last of all a quarter period of silence. Now believe it or not the sound produced is the same as the original speech. This is of course only true as long as there is a stage of filtering after the MM54104. The reason is because of the coding method used for the speech data which introduces a high frequency pre-emphasis. It is necessary to have a low pass filter with a cut off frequency of approximately 200Hz and attenuation characteristic of 20dB per decade. The cut off point may be varied according to the nature of the voice being synthesised eg 100Hz for a man's, but maybe 300Hz for women's and children's. **Fig 8** shows the minimum low pass filter plus amplifier used with the chip. In some applications there will be a requirement for an additional filter to limit frequencies above 3kHz. **Fig 9** shows a complete filtering network which will produce the ideal frequency response for the system.

Nothing has been mentioned so far about the data rate. It is in fact on average 1000 bits/sec for a male voice and little more for females and children. Even though this is rather higher than that of the allophone synthesiser, it is still commendably low. It has been achieved by the compression techniques used in the speech analysis program, particularly the periods of silence where no data is required

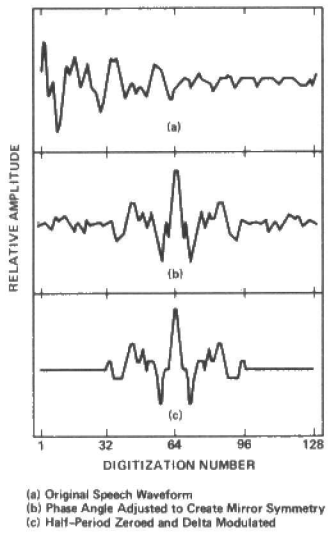
and the repeating, in reverse, of sequences of speech waveforms.

The "Digitalker" system is capable of producing very good speech realism indeed, particularly so if whole phrases are synthesised. This realism allows the sex of the individual as well as regional accents to be recognised. The only real drawback, which may not bother many users, is the limitation on size of vocabulary.

Predictable Speech

Yet another system of speech synthesis which has been adopted by companies like AMI, Hitachi and Texas Instruments is that of Linear Predictive Coding (LPC). So what is this LPC one might ask. The story starts back at the human voice tract again. During the act of speaking the spectral characteristics of the vocal chambers change with time. The change is, because of things like speed of muscle movement, not particularly fast. Knowledge of this and of human perception of speech, allows a model of the vocal tract to be constructed that only requires to be supplied with speech parameter data every 20-25 milliseconds. The data obtained when a speech waveform is sampled at a certain time interval called the Nyquist interval, bears a strong correlation to the data obtained in adjacent intervals. To elaborate, it is possible to predict the speech parameter data for a coming speech sample from a weighted linear combination of the data from preceeding speech samples, hence the term Linear Predictive Coding.

The model of the vocal tract is usually visualised as a series of 10 equal length sections of tube, each with a different diameter as in **Fig 10.** Every one of the sections has a particular filter co-efficient assigned to it. At the points where the cylinders connect reflections occur. These reflections, caused by impedence mismatching, contribute to the production of resonance. In each of the sections of



(a) Original Speech Waveform
(b) Phase Angle Adjusted to Create Mirror Symmetry
(c) Half-Period Zeroed and Delta Modulated

Figure 5. An illustration of what happens to the waveform of a typical voiced pitch period during its conversion.

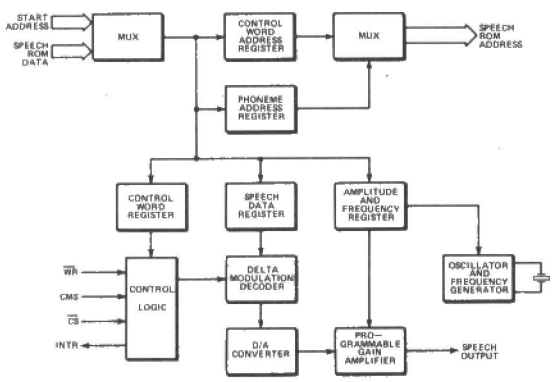


Figure 6. Block diagram of National's SPC (Speech Processing Chip) that forms the heart of their Digitalker system.

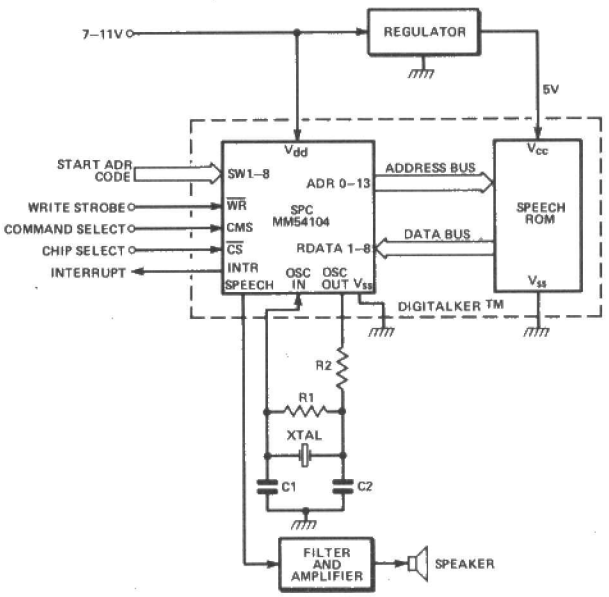


Figure 7. This diagram shows how few components are necessary to realise a minimum configuration Digitalker system.

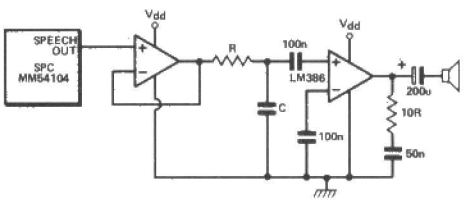


Figure 8. The output from the SPC chip must be low pass filtered by a straightforward circuit based on very few components.

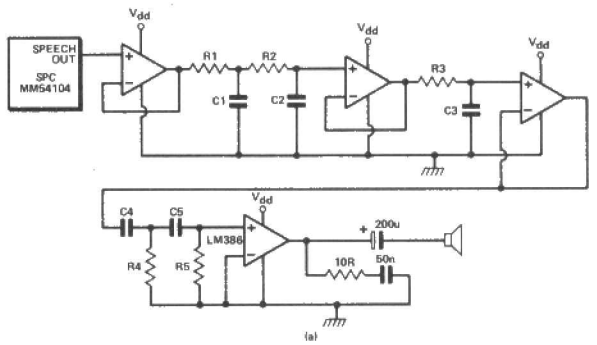


Figure 9(a). This circuit shows a more sophisticated filter which will produce the ideal frequency for the Digitalker system. This augments the basic high pass circuit by providing additional filtering to limit frequencies above 3kHz.

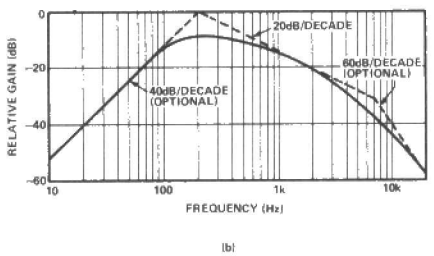


Figure 9(b). A plot of frequency vs relative gain for the circuit of Fig. 9(a).

the tube the fraction of the progressive waves determined by the reflection co-efficient 'k' for that particular section, is added to the regressive waves and vice versa. In an LPC synthesiser the calculation is carried out by a 10-stage digital filter which has 'k' parameter data fed to it. In addition, it must be remembered that the shape of the human vocal tract changes slowly with time and this too is modelled into the digital filter where the 'k' reflection co-efficients will also alter with a certain time period so that the filter continues to faithfully emulate the changes in the vocal tract. In addition to the 'k' parameters, the filter needs to be fed with data concerning the pitch of the voice and its loudness. The time period at which all this is done is called a data frame and as previously said it needs updating at 20-25 millisecond intervals (Hitachi do it very 10 or 20ms).

Each data frame may contain up to approximately 49 bits which means an average data rate of upwards of 2400 bits/sec. In practice this is reduced down to an average of 1200 bits/sec by the following factors:-

1. When the speech data for successive frames is very similar it can be repeated without change. It is often possible to do this because of the slowly changing changing shape of the vocal tract.
2. Unvoiced sounds require fewer reflection co-efficients.
3. When there are pauses in the speech no co-efficients are required.

In terms of memory size this data rate means that 20 secs of speech or 24,000 bits can be available in one IC. When whole phrases and sentences are analysed before recording into memory for subsequent synthesis the LPC system preserves, to a large degree, the natural inflections and intonations in the speech. This is good news for those situations that demand high quality renditions.

It is now time to have a look in detail at a Texas Instruments Voice Synthesis Processor (VSP). This is the TMS 5220 P-channel MOS chip. It has a number of interesting features which make it easy to use. Firstly, virtually all the I/O and processing operations necessary for speech production are handled by the VSP itself. This means that it looks to the controlling CPU like a simple periferal and so speech capability can be added to an existing microprocessing system without worrying about overloading it. Having said that it is probably a good time to fasten your safety belts for a look at the internal workings of the chip to see how it is all done.

Figure 11 shows a block diagram of the VSP. As can be seen, there are a number of sections the workings of which will obviously need a little explaining. The best place to start is probably the timing section around which all the action takes place. In this section there is a 640kHz RC oscillator that has its output divided by 4 to produce two major phases called PH-1 and PH-2, with corresponding precharge clocks PH-3 and PH-4 (**Fig 12** explains the relationships between all these). All the control and timing operations of the chip occur on one of the 6.25 microsecond major phases. One 6.25 microsecond is a bit time and there are 20 bits per sample period (ie an 8kHz sample rate). Twenty-five sample periods make up a 3.125 millisecond interpoplation interval and 8 of these go to make the 25 millisecond frame period. It is during the first of these interpolation periods that new speech data is transferred to the synthesiser. Now on to the CPU interface.

It consists of an 8-bit bidirectional data bus (D0-D7), read and write lines (RS and WS), a ready line (READY) for keeping things synchronised and an interrupt line (INT) to call the CPU's attention if it is required to do something like providing new data. In addition to receiving speech data from the CPU the VSP has access to as many as 16 Texas Instruments TMS 6100s (128k-bit serial ROMS). Communication takes place between the VSP and the 6100s by means of a 4-bit parallel data bus (ADD1,2,4,8), two control lines (M0,M1) and the clock (ROMCLOCK).

The COMMAND register holds command data it receives from the Memory Bus for the controller to interpret and execute. This allows the VSP to behave as an attached processor to the host CPU and to carry out its synthesising all by itself when appropriate commands are sent by the host CPU.

The FIFO Buffer is used to hold speech data coming in from the CPU prior to its being used for synthesis, in addition, the buffer signals to the CPU as it empties that more speech data is required.

The STATUS register holds data that informs the CPU of the current state of the VSP while the DATA register is used to formulate a byte of data from the serial data entering from the Voice Synthesis Memory (VSM).

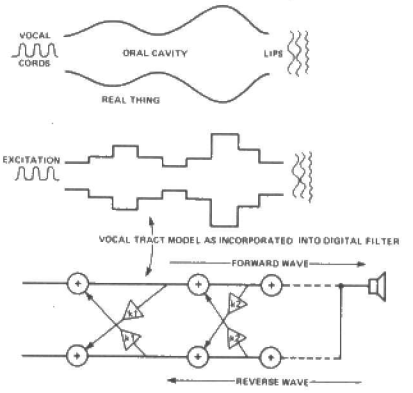


Figure 10. The vocal tract may be visualised as a series of 10 equal length sections of tube each of different diameter.

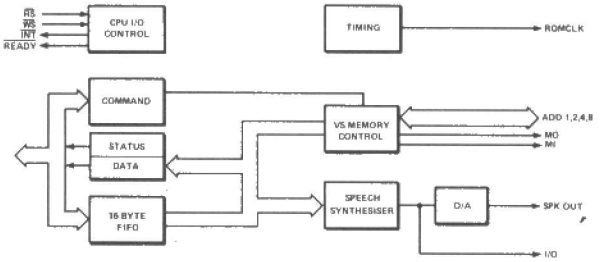


Figure 11. Block diagram of the TI Voice Synthesis Processor (VSP). This device is straightforward to use by virtue of the fact that most of the I/O and processing operation necessary for speech production are handled by the VSP itself.

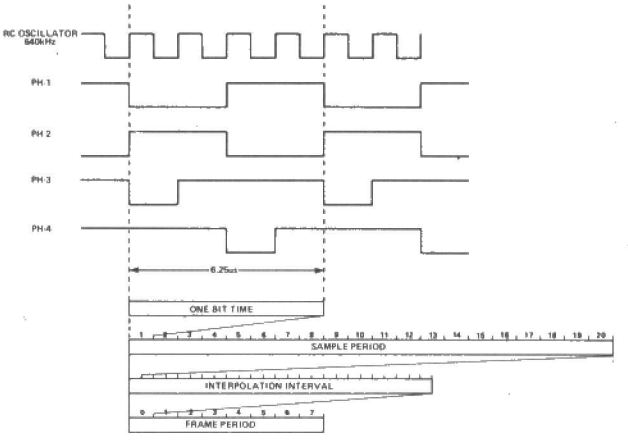


Figure 12. The relationships between the clocks derived from the 640kHz RC oscillator of the VSP.

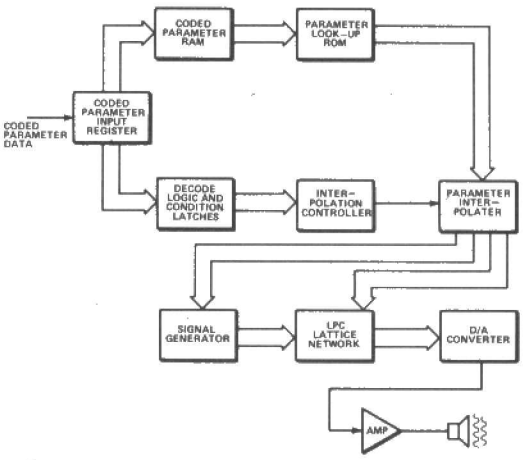


Figure 13. A simplified block diagram of the Speech Synthesiser indicating the way in which LPC data is reconstituted into speech.

A simplified block diagram of the Speech Synthesiser section is shown in **Fig 13** and we are now to look at how LPC data is changed into speech.

The PARAMETER INPUT REGISTER receives data either from the memory ICs or from the host CPU via the FIFO buffer. This data is then unpacked as it were and tested to make sure it is all right. After this the coded parameter data is stored in the CODED PARAMETER RAM from which it will be used to obtain the appropriate parameter value from the PARAMETER LOOK-UP ROM. This data is then used by the processing logic of the PARAMETER INTERPOLATOR as the target to reach for the new frame from the data it had in the previous frame period and in order to facilitate this action there is a 50-bit RAM available for storing the data. It is in between the frame periods that the logic updates the model of the vocal tract with new filter, pitch and energy parameters eight times or every 3 milliseconds approximately. Thus the new pitch and energy parameters are sent to the SIGNAL GENERATOR that excites the digital filter and the new 'k' values are passed to the 14-bit LPC LATTICE NETWORK. This means that at the end of each sample period the 8-bit D/A CONVERTOR is presented with a new set of data to change into speech. It is because of this update that a smoothly varying audio output is finally obtained.

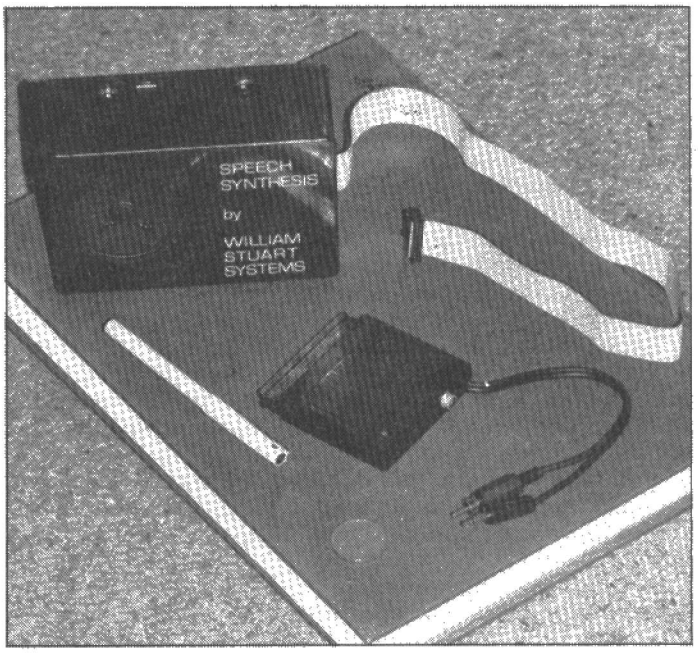
The lattice filter output is sampled every 125 milliseconds and of the ten MSB the seven low-order bits and the MSB (sign bit) are passed to the D/A convertor for changing into speech while the other two bits are logically added to the sign bit and used to put the driver either fully on or fully off. The audio output of the D/A convertor is current sourced and designed to interface directly with an SN76489AN audio amplifier.

The Last Word

These articles have shown that there are a wide variety of approaches to the problem of electronic speech generation and that the various systems all involve trade offs between various parameters such as memory overhead and speech quality. With memory becoming cheaper and denser with each year the likelihood is that within a couple of years capable of generating human quality speech from a large vocabulary base will be commonplace.

Added to the developments taking place in the complementary area of speech recognition, it will not be long before one can hold a conversation with a computer. The challenge then will be to the software producers to come up with programs that will be able to cope with the vagaries of human grammer.

E&CM



Two commercial systems that have adopted GI's allophone based system – William Stuart Systems synthesiser and the new μspeech from Currah.

SINGLE CHIP MICROCONTROLLER

Part 3

Richard Whitlock completes his description of Motorola's 68705P3 self programming microcontroller.

In the past two issues of *E&CM* some of the features of the 68705P3 have been discussed along with details of a self programming board designed for use with the device. This month we round off the description of the MCU itself before moving on to a look at a cross assembler for the MCU in our January issue.

Ports A, B & C

Ports A, B and C are programmable, under software control, to be either inputs or outputs on a line by line basis. The signal direction is controlled by the logic value of the corresponding bit of the port's associated Data Direction Register (DDR). A port line is an input if the corresponding DDR bit is a logic "0" and an output if the DDR bit is a logic "1". On Reset all bits of all DDRs are cleared, putting all port lines into the input mode. However the port output data registers are not initialised to any known state and so should be loaded with good data before the DDRs are configured to allow output on any lines.

All DDRs are write only registers and read as all "1s" regardless of the logic levels last written into them. This prohibits the use of any read/modify/write type of instruction to manipulate their contents. Simple byte store instructions should be used to configure them.

When lines are configured as outputs the bits read from their port address reflect the data stored in the data output register not the actual voltage levels on the external port pins, which are prey to the effects of circuit loading.

No special control lines exist alongside the data lines, there is no input data latching and no "hardware hand-shake" circuitry is provided. All lines of all three ports are TTL compatible as both inputs and outputs.

Address and special characteristics details for each port are given in **Table 1** (below).

Interrupts

All three 68705 variants have provision for three sources of interrupts. The two 40-pin versions also have a fourth possible interrupt channel, which is to be discussed in a later issue of *E&CM*. The three common to all 68705s are as follows: THE EXTERNAL INTERRUPT INPUT – (INT) – This input incorporates Schmitt trigger circuitry allowing it to register both fast edged digital signals and slow sinusoidal signals' zero-crossings. In either case it is the negative going transition that triggers the interrupt. The maximum frequency of a signal that will be recognised by the internal logic connected to the INT pin is exactly the same as that vien above for the TIMER input, though of course this limit takes no account of the time required to service the series of interrupt requests generated.

Figure 1 shows the recommended input circuits for both digital and sinusoidal signals. The ability to interface to a low voltage AC signal (eg from the logic power supply transformer) has important applications in machine control, where switching of heavy inductive loads can be synchronised with the mains waveform to minimise RFI emissions and contact erosion.

The external interrupt channel via the INT pin can only be masked by setting the main interrupt mask bit, I, in the CCR. Thus, when this source of interrupts is disabled, all hardware interrupts are disabled. Masked interrupt requests are latched until such time as the I-bit is cleared, when they take effect. If two or more interrupts from the same source occur during the time that the I-bit is set, only the last will be recognised.

When the I-bit is set the processor can still sense the signal level on the INT pin and take appropriate action by means of two special conditional branch instructions, which reference the logic level of the INT pin Schmitt trigger output, rather than that of a CCR bit.

PORT A – Data Register address: 000H. DDR address: 004H.
An 8-bit wide port on all 68705 variants.
CMOS compatible as OUTPUTS only (due to the presence of resistive pull-up elements).

PORT B – Data Register address: 001H. DDR address: 005H.
An 8-bit wide port on all 68705 variants.
CMOS compatible as INPUTS only.
As outputs these lines may each sink 10mA at a low enough drain voltage to drive LEDs powered from the 5V line via a current limiting resistor and source 1mA at about 1.5V to drive Darlington transistor switches.

PORT C – Data Register address: 002H. DDR address: 006H.
An 8-bit wide port on the 68705U3 and 68705R3, but only a 4-bit wide port on the 68705P3, where the unimplemented bits read as all "1s".
CMOS compatible as INPUTS only.

THE TIMER INTERRUPT – As noted above in the Timer section, the Timer will, if its local interrupt mask bit (TIM in the TCR) is clear, pass an interrupt request to the processor when the main counter decrements through zero. This interrupt request will only be recognised if the I-bit is clear.

THE SOFTWARE INTERRUPT INSTRUCTION – (SWI) – This form of interrupt is not maskable and is always executed when encountered in a program, just like any other instruction. It is often used as a break-point during program development, allowing a particular, externally observable, operation to be called for execution to mark the fact that the processor has arrived at a particular place in the main program, an event that would be hard to detect in a one-chip system otherwise, since the user has no access to the system's bus lines to detect address line levels etc.

When an unmasked interrupt request reaches the processor, the execution of the current instruction is completed before any other action is taken. When the PC has been incremented to point to the next opcode, main program operations are suspended and the processor spends the next 11 cycles stacking its internal registers, setting the I-bit in the CC register, to mask further interrupt requests, and fetching the appropriate interrupt service routine vector from the top of memory space. The service routine vector storage locations for the various types of interrupt are given in **Table 2.**

When handling multiple sources of interrupt in an application program, it must be remembered that if another, higher priority, interrupt is to be allowed to interrupt the service of the first occuring interrupt, the I-bit must be cleared at the beginning of the service routine or the second interrupt will not be recognised until the old processor CC register contents are restored at the return to the main program.

The priorities of the various sources of interrupts are given in **Table 2.** If the I-bit is clear, the external interrupt, via the INT pin, has the highest priority, followed by the Timer interrupt, and either of these will divert the processor from fetching the next instruction in the main program even if it is an SWI. If the I-bit is set, only the SWI instruction is capable of interrupting the processor, it thus has the default priority of 2, priority 1 always being held by the Reset signal.

The Master Oscillator

The internal master oscillator circuitry is designed to offer great flexibility while requiring the minimum of external components. As noted in the MOR section last month, two groups of external frequency

control circuits are allowed, selection between these being made by programming MOR bit 7 (CLK) to be either logic "1" or logic "0".

If the CLK bit is logic "1", two low cost external circuits are available (see **Fig 2c and d**), using just a wire or PCB link or a pull-up resistor. A resistor selection plot is reproduced in **Fig 3** for use with the circuit in **Fig 2d**).

If the CLK bit is logic "0", either an external crystal or an externally generated

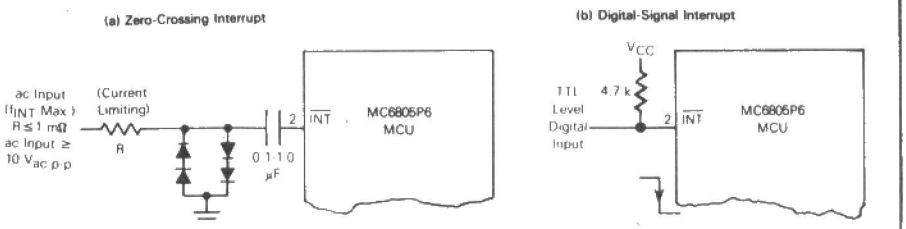


Figure 1. Recommended input circuits for digital and sinusoidal signals.

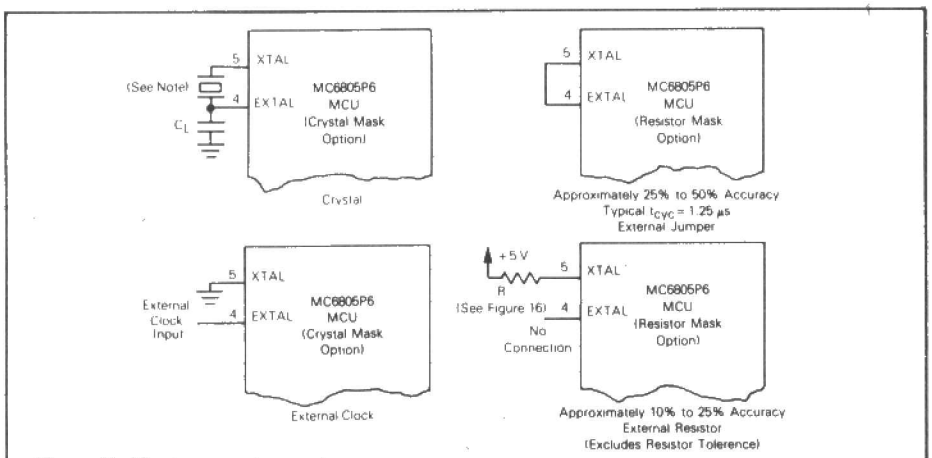


Figure 2. Clock generator options.

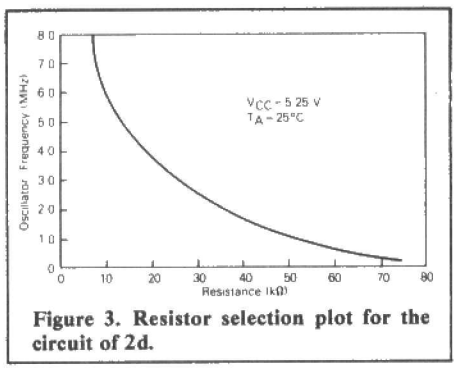


Figure 3. Resistor selection plot for the circuit of 2d.

TTL level clock signal may be used (see **Fig 2a & b**). In the external crystal circuit the recommended load capacitance (C_L) is 27p for a 4MHz crystal, a value that appears to work well with a 3.58MHz colour – burst crystal. However if a much lower crystal controlled frequency is required, a second load capacitance (C_L) will also be required for stable operation.

Stray capacitance due to circuit board layout can be a problem delaying the establishment of stable oscillation on power-up. If the problem is not too severe it can be rectified by increasing the value of the delay capacitor connected to the RESET pin. The author has increased this to 2u from the 1u recommended in the device data sheets without apparent ill-effects, clearing up oscillator start-up problems in the process. If this remedy does not work there is no option left but to redesign the PCB. Two recommended layouts for the crystal and load capacitors are given in **Fig. 4.**

It appears that, in common with other Motorola microprocessors, the 68705s have their processor registers implemented as dynamic storage registers, which require refresh operations even more frequently than does normal dynamic RAM. This requirement imposes a lower limit on the master oscillator frequency used. 400kHz is the lowest master oscillator frequency that is guaranteed to retain data in all chips released for sale. This frequency is divided by four to generate the processor clock whose period, t_{cyc}, is the basic measuring interval for instruction execution time and external signal compatability calculations. The highest master oscillator frequency that the chip can cope with is 4.2MHz, which allows a margin of safety when a standard 4MHz crystal is used. The upper and lower limits for the processor clock frequency are thus set at 1MHz and 100kHz respectively.

TABLE 2

INTERRUPT & RESET VECTORS

Interrupt	Priority	Vector Address
$\overline{\text{RESET}}$	1	$FFE and $FFF
SW1	2*	$FFC and $FFD
$\overline{\text{INT}}$	3	$FFA and $FFB
TIMER/$\overline{\text{INT2}}$	4	$FF8 and $FF9

* Priority 2 applies only when the I-bit in the Condition Code Register is set (as when a service routine is occurring). When I = 0 and all interrupts are being accepted. SW1 has a priority of 4 (like any other instruction). The priority of $\overline{\text{INT}}$ thus becomes 2 and the timer becomes 3

RESETS

The internal circuitry connected to the RESET pin includes a resistive pull-up and a Schmitt trigger input to further logic. On powering up the IC, an external capacitor connected to the RESET pin is charged via the pull-up resistance until the voltage on the RESET pin reaches the high-going threshold voltage of the Schmitt trigger, at which point the Schmitt trigger's output changes state,

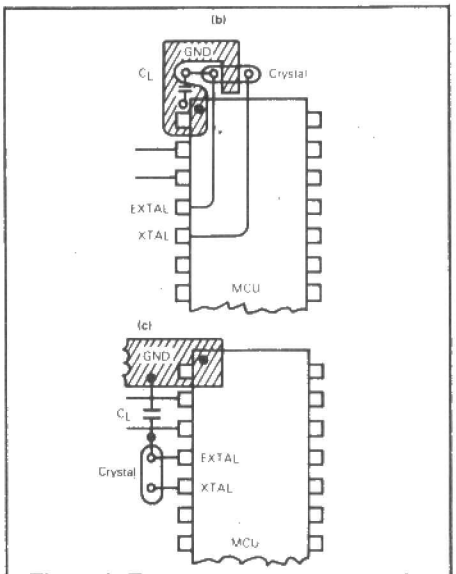


Figure 4. Two recommended layouts for crystal and load capacitors.

removing the internal Reset signal that has been asserted up to that point. The delay provided by the charging of a 1u capacitor is ideally sufficient to allow the master oscillator to stabilise before removing Reset, but, in the event of slow oscillator start-up, this capacitor can be increased to 2u, and perhaps beyond, without any ill-effects being detectable.

For a "warm-start" Reset, the capacitor can be grounded by either a manual switch or an open collector TTL output. The voltage on the RESET pin must be pulled below the Schmitt trigger low-going threshold, (about 1.6V), for a period longer than one processor clock cycle, t_{cyc}, for the Reset to be effective. There appears to be no practical limit to the length of time that the processor may be held in the Reset state, nor any need to synchronise the removel of Reset to any other event.

When a Reset is recognised during normal operation, the current instruction is not completed before the program is aborted. The SP is reset to 07FH, the I-bit in the CC register is set, any pending interrupts are cleared, the main down counter of the Timer is set to all "1s", as is the Prescaler, all DDRs are cleared, all secondary interrupt mask bits are set, and the remaining bits of the TCR are also all set. When the internal Reset signal becomes inactive, the contents of the MOR are loaded into their destinations in the oscillator circuitry and the TCR before the Reset vector is fetched from the top of memory and loaded into the PC. Program execution then commences from the location pointed to by the PC.

E&CM

MICROTAN 65

A versatile 6502 based processor board presented in conjunction with Microtanic Computer Systems.

The market today is swamped with general purpose computer systems that, with a few notable exceptions, place more emphasis on 'bells and whistles' features such as colour displays and zappy sound generations than on any sensible I/O structure. Fine for playing games, but not of much use in more serious development/control applications.

The single board computer described here, and in subsequent issues, is designed for the latter applications and consists of a fully expandable 6502 based system.

This month we present an overview of the system together with the board's circuit diagram. In subsequent installments we'll describe detailed circuit operation, construction of the board and possible areas of application.

Design Outlines

It was considered that the majority of users of a single board computer would initially benefit from a system that was easy to use rather than one that has parallel I/O but was difficult to use. The board therefore incorporates an on-board VDU and this allows an alphanumeric keyboard to be used from the word go.

The 6502 microprocessor was chosen mainly for its very simple yet elegant hardware structure. Of course, as most readers will know, the 6502 also has a very powerful instruction set and addressing modes.

A fully expanded system contains the 6502 microprocessor, 1K ROM containing TANBUG, 1K RAM which is used for display memory and user program, keyboard interface, VDU logic and tri-state address buffers.

The most important aspect of the design is that the address multiplexer is switching at the processor clock rate. This can be done with the 6502 as memory accesses will only occur on one phase of the clock ie when 02 is high. When 02 is low the memory is not selected. During this time the VDU logic reads the display memory, one location at a time and decodes memory contents as alphanumeric or graphics characters. This means that the computer's display is free from annoying speckles, spots and flashes. This is because there is no conflict of access to display memory between processor and display refesh logic. In fact it is possible to run a program that resides in display memory and it will run at full speed without upsetting the display!

Memory mapped display: Pages 2 and 3 of the address space are used as the display memory, that is from hex location 200 to hex 3FF. Location 200 is the top left hand corner character position. Location 201 is the second character position in the top row and so on.

To write a particular character into a character cell in the display the user must write the ASCII code, or whatever, for that character in the correct memory location. For example, if it was required to write the character "W" in the bottom row third column then the ASCII code 57 (for W) should be placed in memory location 3E2. If using TANBUG's memory modify command M to load characters into the display remember that the display scrolls automatically and the character may be output onto a different row than intended.

Memory mapping of the system is controlled on-board by three wire links, when used without the TANEX expansion board. As there is only 1K RAM, 1K ROM and 6 I/O locations on the microtan 65 each of the three are allowed to repeat through defined areas of the 64K address space to simplify design.

As can be seen the address space is divided into only three blocks and therefore only two bits of the address bus are necessary to control this map. The two bits used for memory mapping are the most significant address bits A15 and A14. Three links are used to wire in this address map on the system. When TANEX is used however, all memory mapping is controlled on the TANEX board and not on the Microtan 65. To modify the Microtan 65 it is only necessary to cut these three links.

RAM and ROM hardware is of really no consequence to the user but a knowledge of the on-board I/O ports is useful. There are six peripheral ports used on the CPU board and these control graphics on and off, keyboard read and write, keyboard interrupt flag clear and delayed non-maskable interrupt (used for single instruction and breakpoints). Each one of these will be taken in turn:

(a) Graphics on and off: The display refresh controller reads the display memory a byte at a time and interprets the bottom seven bits of each byte to be an ASCII coded character. There is however a ninth bit which can only be written to by the CPU. This ninth bit comes from the 2102 1K x 1 bit RAM chip in the graphics option. If at a certain display location the ninth bit, or graphics bit, is a logic one the data byte read is not decoded as an ASCII character but is instead used to build up a graphics block of 2 x 4 pixels, in the character cell position.

The graphics on and graphics off I/O ports control an R-S flip-flop, the output of which is the data input to the 2102 RAM chip. Therefore if the graphics flip-flop is in the on position each character written to the display memory will be decoded as a graphics character. Because the CPU cannot read the graphics bit directly it is not possible to scroll displays containing graphics. As displays containing graphics are rarely scrolled anyway this should not prove to be too much of a handicap.

The operations required to write graphics characters to the display is to firstly turn the graphics on by executing the assembly code instruction LDA BFF0. Each character then written to display will be a graphics character until graphics are turned off by executing the assembly code instruction STA BFF3.

(b) Keyboard read, write and interrupt clear: There is both an input and output port on the keyboard socket. The output port is used only for strobing the 20 key keypad. This output port is used by executing the assembly code instruction STA BFF2. The input port is used for both types of keyboard. When using the alphanumeric keyboard the strobe from the keyboard is used to clock a flip-flop (the keyboard interrupt flip-flop). The output of this flip-flop is the keyboard interrupt flag and forms the eighth input bit to the keyboard input port. If interrupts are enabled in the CPU then an interrupt will be generated by the keyboard strobe. Reading the keyboard input port will then allow the CPU to test whether or not the interrupt was generated by the keyboard or some other device. If interrupts are not enabled then the keyboard interrupt flag will still be set but will not generate an interrupt. It is possible to read the flag by reading the keyboard input port, thus a strobe from an alphanumeric keyboard can be detected either by an interrupt or by software polling. Whichever technique is used though it will be necessary to reset the keyboard interrupt flag after it has been set and detected in order that it is ready to register a further keyboard strobe. To reset the keyboard interrupt flag the assembly instruction STA BFF0 should be executed. The keyboard port is read by the instruction LDA BFF3.

(c) Delayed non-maskable interrupt: This facility is used by the monitor program TANBUG when executing single instructions and breakpoints. It should not be used in a user program. Users who will contemplate using the non-maskable interrupt in special applications will probably not be using TANBUG. In these cases the link LKNMI should be broken and the non-maskable interrupt may be driven via the board connector.

Circuit Diagram

Just to whet your appetite, **Fig 1** shows the full circuit diagram of the system. Next month we'll examine this in detail and describe the construction of the system.

The single board 6502 system is available as a kit from MCS, at 16 Upland Road, Dulwich, London SE22.

The kit costs £79.35 plus £1.50 p&p and comes complete with a comprehensive set of documentation.

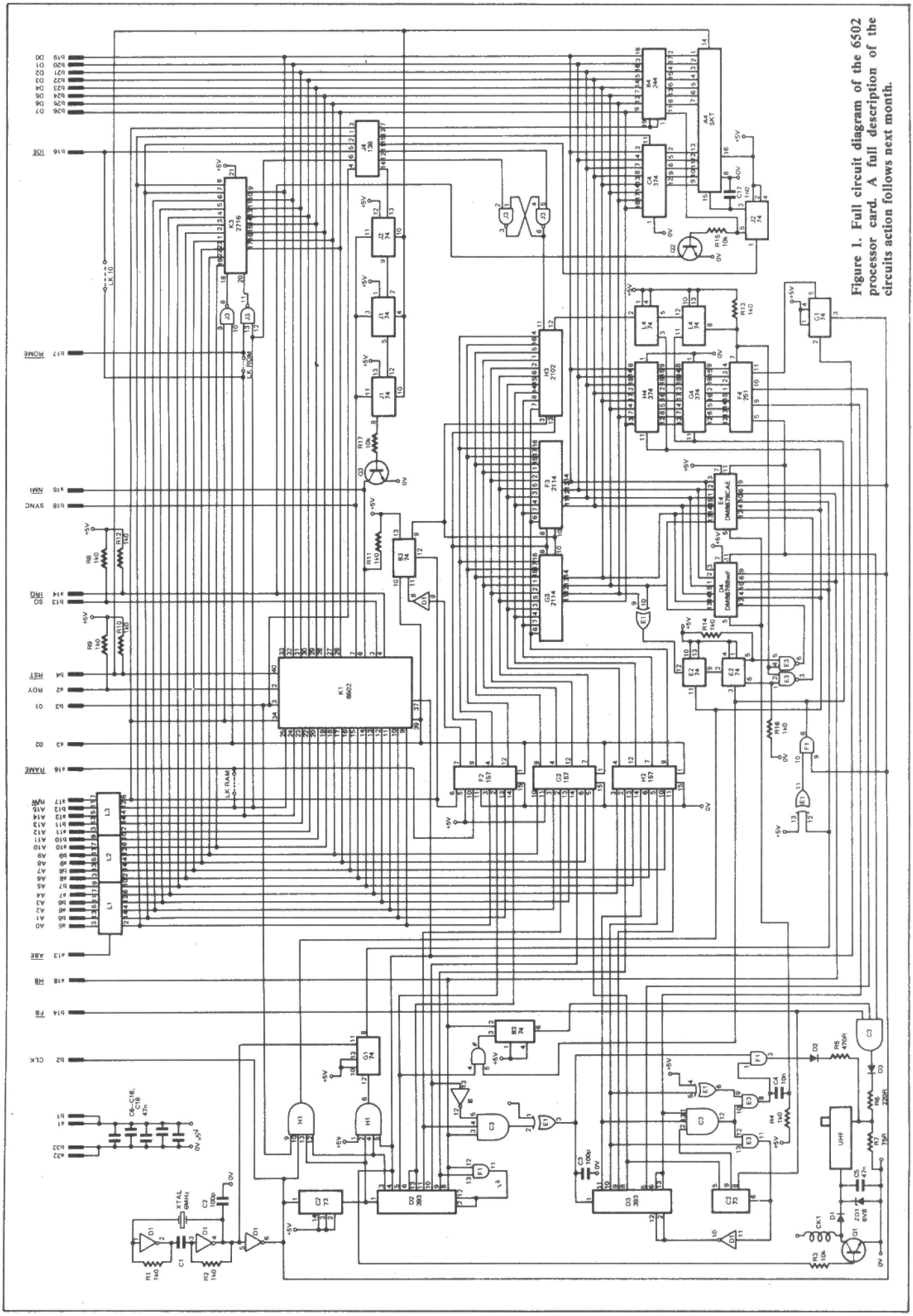


Figure 1. Full circuit diagram of the 6502 processor card. A full description of the circuits action follows next month.

If you're still staggering through the computer jungle and not getting sensible answers to your questions, we have some good news:

Now you can turn to **professional** people who are capable of giving you sound advice on practically all aspects of popular computing.

They all have one thing in common:

They are **COMPUTERS FOR ALL** dealers.

A Computers for All dealer is different from the normal Computer retailer.

Not surprisingly he won't try to sell you things like cameras or cosmetics; stationery or sealing wax.

He will, however, be capable of answering sensibly almost any question you have on computers and computing, and have readily available a wide range of popular computers, hardware, software, books, and peripherals.

So why not call in at your local **COMPUTERS FOR ALL** dealer today?

He can lead you in the direction you want to travel.

The shops where people matter:

AVON

MERCATOR COMPUTER SYSTEMS
3 Whiteladies Road, Clifton, Bristol. 0272 731079.

BERKSHIRE

KENNETH WARD COMPUTERS
Verve House, London Road, Sunningdale. 0990 25025.

CHANNEL ISLANDS

MEGA LTD.
7 Anley Street, St. Helier, Jersey. 0534 72263.

MICROTEL
Camp-Du-Roi Vale, Guernsey. C.I. 0481 53436.

CHESHIRE

THE COMPUTER CENTRE
68 Chestergate, Macclesfield. 0625 618827.

CORNWALL

COMPUTAVISION
4 Market St. St. Austell, Cornwall. 0726 5297.

FAL-SOFT COMPUTERS
8 St. George's Arcade, Falmouth TR11 3DH. 0326 314663.

DEVON

A & D COMPUTERS
"Computerland" 6 City Arcade, Fore Street, Exeter. 0392 77117.

BITS AND BYTES
44 Fore Street, Ilfracombe, N. Devon EX34 9JD. 0271 62801.

COMPUTER SYSTEMS (TORBAY)
Pump Street, Brixham. 08045 6565/6.

CRYSTAL COMPUTERS
209 Union Street, Torquay. 0803 22699.

SYNTAX LTD.
Midhurst, Grenofen, Tavistock, Devon. 0822 2392.

DORSET

DENSHAM COMPUTERS
329 Ashley Road, Parkstone, Poole. 0202 737493.

ROBERTS ELECTRICAL (BLANDFORD) LTD.
14 Market Place, Blandford Forum, Dorset. 0258-55898.

ESSEX

AKHTER INSTRUMENTS
Unit 19, Arlinghyde Estates, South Road, Harlow. 0279 412639.

CHELMSFORD COMPUTER CENTRE
3 Baddow Road, Chelmsford. 0245 356834.

COMPUTERAMA
88-90 London Road, Southend-on-Sea, Essex. 0702 335443.

COMPUTERS FOR ALL
72 North Street, Romford. 0708 752862.

HANTS

MICRO VIDEO STUDIOS LTD.
60 Normandy Street, Alton, Hants. 0420 82055.

HERTS

VIDEO CITY
45-47 Fishers Green Road, Stevenage. 0438 53808.

THE COMBINED TRADING CO.
10 & 11 Salisbury Square, Old Hatfield. 07072 65551.

KENT

ANIROG COMPUTERS
29 West Hill, Dartford. 0322 92513.

APHROS SOFTWARE CO.
47 Hawley Square, Margate. 0843 294699.

DATA STORE
6 Chatterton Rd., Bromley, Kent. 01-460 8991.

JUTEA LTD.
92 High Street, Bridge, Canterbury, Kent. 0227 831183.

LEAKES LTD.
63 Sidcup High Street, Sidcup, Kent. 01-300 1142.

LANCS

P.C.S.
39 Railway Road, Darwen. 0254 776677.

4MAT COMPUTING
67 Friargate, Preston. 0772 561952.

SUMLOCK
Royal London House, 198 Deansgate, Manchester M3 3NE. 061 834 4233.

LEICS

DIMENSION
27-29 High Street, Leicester. 0533 57479.

MOVIES COMPUTER CENTRE
5 Church Street, Melton Mowbray. 0664 61169.

LONDON

EROL COMPUTING
125 High Street, Walthamstow, London E17. 01-520 7763.

KAYDE HOME COMPUTERS
1 Station Approach, New Eltham, London SE9. 01-859 7505.

KELLY'S COMPUTERMARKET
227 Dartmouth Road, Sydenham, London SE26 4QY 01-699 4399/6202.

MICRO ANSWERS LTD.
70-71 Wilton Road, Victoria, London SW1. 01-630 5995.

MIDDLESEX

SCREENS MICROCOMPUTERS
6 Main Avenue, Moor Park, Northwood. 09274 20664

TWILLSTAR COMPUTERS
17 Regina Road, Southall 01-574 5271.

N. IRELAND

D. V. MARTIN LTD.
13 Bridge Street, Belfast. BT1 1LT. 0232 226434.

OXFORDSHIRE

SCIENCE STUDIO
7 Little Clarendon, Oxford OX1 2HP. 0865 54022.

SURREY

ANIROG COMPUTERS
8 High Street, Horley. 02934 6083.

COMPUTASOLVE
8 Central Parade, St. Marks Hill, Surbiton. 01-390 5135.

KWX BUSINESS SUPPLIER
Old Coach House, Belmont, Sutton, Surrey. 01-642 9471.

SNOW-BEECH
1 East Grinstead Road, Lingford, Surrey. 0342 832476.

THE COMPUTER-WAY LTD.
738 North Street, Guildford, Surrey. 0483 62626

SUSSEX

WORTHING COMPUTERS
32 Liverpool Road, Worthing, W. Sussex. 0903 210861.

WALES

MICRO-CARE COMPUTING LTD.
18 Baneswell Rd., Newport, Gwent NPT 4BP. 0633 50482.

TRYFAN COMPUTERS LTD.
57 Madoc Street, Llandudno, Gwynedd. 0492 70802.

TYNE & WEAR

THE COMPUSHOP
10 Newgate Centre, Newcastle-upon-Tyne NE1 5RE. 0632 618673.

WARWICKSHIRE

IMPULSE MICRO SYSTEMS LTD.
6 Central Chambers, Cooks Alley, Wood Street, Stratford-Upon-Avon. 0789 295819.

W. MIDLANDS

CALISTO COMPUTERS
119 John Bright Street, Birmingham B1 1BE. 021-632 6458

JBC
200 Earlsdon Ave. North, Earlsdon, Coventry. 0203 73813.

WORCESTERSHIRE

EVESHAM COMPUTER CENTRE
Crown Court Yard, Bridge St., Evesham. 0386 48635.

YORKSHIRE

COM-TEC
6 Eastgate, Barnsley, South Yorkshire. 0226 46972.

EMPIRE ELECTRO CENTRES
783-789 Leeds Road, Bradford BD3 8BY. 0274 662476.

HARROGATE COMPUTER CENTRE
18 Cheltenham Parade, Harrogate, N. Yorkshire 04 23 57126.

TOWERLIGHT LTD.
7 Crown Street, Hebden Bridge, W. York HX7 8EH. 0422 843520.

COMPUTERS FOR ALL
DEALERS